
# PCI Standard Hot-Plug Controller and Subsystem Specification

**Revision 1.0**

**June 20, 2001**

| REVISION | REVISION HISTORY | DATE |
|---|---|---|
| 1.0 | Initial release. | 06/20/01 |

The PCI Special Interest Group (SIG) disclaims all warranties and liability for the use of this document and the information contained herein and assumes no responsibility for any errors that may appear in this document, nor does the PCI SIG make a commitment to update the information contained herein.

Contact the PCI SIG office to obtain the latest revision of the specification.

Questions regarding the PCI Standard Hot-Plug Controller and Subsystem Specification or membership in the PCI SIG may be forwarded to:

**Membership Services**
www.pcisig.com
E-mail: administration@pcisig.com
Phone: 503-291-2569
Fax: 503-297-1090

**Specification Distribution**
Phone: 1-800-433-5177 (Domestic Only)
503-291-2569 (International)
Fax: 503-297-1090

**Technical Support**
techsupp@pcisig.com

DISCLAIMER

# Contents

# Figures

# Tables

# Preface

The *PCI Hot-Plug Specification, Revision 1.0*, was formally released on October 6, 1997. That specification for the first time established a standard method for inserting and removing PCI add-in cards without powering down the hardware Platform or rebooting the system software. The specification met with immediate commercial success and was introduced in hardware Platform and operating system products from multiple vendors. Since that time, PCI hot-plug technology has become a required feature of industry-standard servers and other high-availability hardware Platforms.

One of the features of *PCI Hot-Plug Specification, Revision 1.0*, that enabled its rapid acceptance was the flexibility it allowed in the implementation of the Hot-Plug Controller. This flexibility allowed each hardware Platform vendor to move quickly to market with hardware and software that met their specific needs.

As PCI hot-plug technology became commonplace, a new opportunity appeared. If more features of the Hot-Plug Controller were standardized, the cost of the hardware could be reduced, and hardware Platform and software development time could be reduced; thereby, reducing the time required to bring each new product to market. Furthermore, a standard implementation would provide a consistent user experience, even if the hardware and software were from different vendors. This, in turn, would reduce customer training and support costs thereby improving customer satisfaction. It is this opportunity that lead to the development of the *PCI Standard Hot-Plug Controller and Subsystem Specification, Revision 1.0*.

# 1. Introduction

## 1.1. Scope

A Hot-Plug Controller is part of a system that conforms to the requirements of *PCI Hot-Plug Specification, Revision 1.1*. As described in that specification, the Hot-Plug Controller turns slots on and off under software control. The Hot-Plug Controller uses slot-specific power switches and signal switches to isolate a slot from the bus for hot-removal or hot-insertion.

The primary purpose of this document is to specify a standard implementation of a PCI Hot-Plug Controller called the Standard Hot-Plug Controller (SHPC). A Platform (as defined in PCI HP 1.1) that includes an SHPC must meet all the requirements of both the *PCI Hot-Plug Specification, Revision 1.1*, and this specification. Implementations of an SHPC are permitted to include other features not described in this specification. However, such features are not permitted to present any software-observable differences in the specified features.

Behavior of a hot-plug system that is assumed by the design of the SHPC is called the Standard Usage Model and is specified in Section 2. The Standard Usage Model assumes minimum capabilities and behavior of the hot-plug Platform and system software.

Requirements for some internal features of the SHPC and recommendations for implementation of its interface to and control of its associated PCI slots are presented in Section 3.

The programming model and register specifications for the SHPC are presented in Section 4.

Section 5 specifies firmware requirements for industry-standard server applications.

Section 6 specifies hardware and software requirements that are unique to PCI hot-plug systems that implement power management, that is, systems that include PCI slots and provide methods for placing the SHPC or other portions of the system in reduced-power states and later returning them to the fully functional, full-power state.

Appendix A is a guide to working with mechanical switches that exhibit switch bounce.

Appendix B provides design checklists for systems that are designed according to this specification.

## 1.2. Reference Documents

The following documents are a part of this specification to the extent specified herein:

*Advanced Configuration and Power Interface, Revision 1.0b* (ACPI 1.0b), February 8, 1999, www.teleport.com/~acpi

*Advanced Configuration and Power Interface, Revision 2.0* (ACPI 2.0), July 27, 2000, www.teleport.com/~acpi

*Developer's Interface Guide for IA-64 Servers (DIG64), Ver. 1.0* (DIG64 1.0), November 1999, www.dig64.org

*PCI Local Bus Specification, Revision 2.2* (PCI 2.2), December 18, 1998, PCI Special Interest Group

*PCI to PCI Bridge Architecture Specification, Revision 1.1* (PCI Bridge 1.1), December 18, 1998, PCI Special Interest Group

*PCI-X Addendum, Revision 1.0a* (PCI-X 1.0a), July 24, 2000, PCI Special Interest Group

*PCI Bus Power Management Interface Specification, Revision 1.1* (PCI PM 1.1), December 18, 1998, PCI Special Interest Group

*PCI Hot-Plug Specification, Revision 1.1* (PCI HP 1.1), June 20, 2001, PCI Special Interest Group

*PCI BIOS Specification, Revision 2.1* (PCI BIOS 2.1), August 26, 1994, PCI Special Interest Group

*Addition of SMBus Interface to Connector* (SMBus ECN), August 9, 2000, PCI Special Interest Group

## 1.3. Documentation Conventions

The following documentation conventions are used in this document:

| | |
|---|---|
| Capitalization | As in PCI 2.2, register names and the names of fields and bits in registers and attributes are presented with the first letter capitalized and the rest lower case, for example, Secondary Bus Configuration register, Controller Command register, Number of Slots field, First Device ID field, Controller Interrupt Pending bit.<br><br>Some other terms are capitalized to distinguish their definition in the context of this document from their common English meaning. These terms are listed in Section 1.4.<br><br>Words not capitalized have their common English meaning. |
| Numbers and number bases | Hexadecimal numbers are written with a lower case “h” suffix, for example, FFFFh, 80h. Hexadecimal numbers larger than four digits are represented with a space dividing each group of four digits, as in 1E FFFF FFFFh.<br><br>Binary numbers are written with a lower case “b” suffix, for example, 1001b, 10b. Binary numbers larger than four digits are represented with a space dividing each group of four digits, as in 1000 0101 0010b.<br><br>All other numbers are decimal. |
| Signal names | Signal names defined in PCI 2.2 and PCI-X 1.0a and signals that are unique to hot-plug systems are indicated with **this bold** font. |
| Buses | As in PCI 2.2, collections of signals that are collectively driven and received are assigned the same signal name with numbers in brackets to indicate the bit or bits affected, for example, **AD[31::00]**, **C/BE[7::4]#**, and **AD[2]**. |

| | |
|---|---|
| Slot-specific signals | Names for signals that are replicated for each hot-plug slot are designated by appending **{n}** to the signal name. References to a slot-specific signal for a single slot use a similar notation using the specific slot number. For example, **RST{n}#** are the slot-specific signals that connect to the **RST#** pins of an unspecified hot-plug slot. **RST{3}#** is the slot-specific signal that connects to the **RST#** pin of slot 3. Similarly, **AD[31::00]{n}** are the slot-specific signals that connect to the **AD[31::00]** pins of an unspecified hot-plug slot, and **C/BE[7::4]{1}#** are the slot-specific signals that connect to the **C/BE[7::4]#** pins of slot 1. |
| Clock frequencies | This specification follows the precedent of PCI 2.2 and PCI-X 1.0a by abbreviating clock frequency notation. For example, 33 1/3 MHz is written "33 MHz," 66 2/3 MHz is written "66 MHz," and 133 1/3 MHz is written "133 MHz." |
| Reference information | Requirements of other specifications appear in various places throughout this document and are marked as reference information. Such references are indicated by the abbreviation "(ref)." Every effort has been made to guarantee that this information accurately reflects the referenced document. However, in case of discrepancy, the original document takes precedence. |

## 1.4. Terms and Abbreviations

This specification uses many terms defined in other PCI specifications. The specification that includes the definition of a term is referenced the first time the term is used by this specification.

The following terms and abbreviations are defined in this specification:

| | |
|---|---|
| **Attention Button** | An optional momentary-contact push button, located adjacent to each hot-plug slot. It is pressed by the user to initiate a hot-insertion or a hot-removal at the slot. |
| **connected power fault** | A slot-specific power fault that occurs while the slot is partially or fully connected to the bus. |
| **disable slot** | The process of disconnecting a slot from the bus and powering it down. A slot must be disabled before an add-in card is physically inserted or removed. |
| **enable slot** | The process of powering up a slot and connecting it to the bus. An add-in card in an enabled slot is ready to be addressed by software after the initialization period specified in PCI 2.2. |
| **host bridge** | A bridge between a CPU/memory subsystem and a PCI bus hierarchy. The existence of a host bridge integrated with a SHPC is indicated to the operating system via ACPI methods or firmware tables, as defined in Section 5.2. The configuration registers for a host bridge and its associated SHPC are accessed in Memory Space, as defined in Section 4.3.2. |
| **Host Bridge Register Block** | The block of memory addresses in a host bridge that includes the HBRB Header and zero or more HBRB Capabilities List Items. |

| | |
|---|---|
| **HBRB Capabilities List Item** | An optional feature of a host bridge that is described and controlled by registers in the Host Bridge Register Block. A host bridge integrated with an SHPC must include exactly one SHPC HBRB Capabilities List Item (see Section 4.3.2.4). |
| **HBRB Header** | The first portion of the Host Bridge Register Block. It provides information about the host bridge and a pointer to the list of HBRB Capabilities List Items. |
| **Hot-Plug Resource Table (HPRT)** | The Hot-Plug Resource Table is a collection of information used by some operating systems, that describes the allocation of system resources in a hot-plug system. |
| **hot-plug segment** | The combination of a single SHPC, the bridge with which it is integrated, the hot-plug slots it controls, and the associated slot-control infrastructure. |
| **hot-plug system** | The combination of system software and a Platform containing at least one hot-plug slot. |
| **isolated power fault** | A slot-specific power fault that occurs while the slot is isolated from the bus. |
| **manually operated retention latch (MRL)** | A manually operated retention mechanism that holds an add-in card in a slot and prevents the user from removing the card. |
| **MRL Sensor** | A switch, optical device, or other type of sensor that reports the position of a slot's MRL to the SHPC. Additional requirements for the MRL Sensor are specified in Section 2.2.3. |
| **MRL Switched Signals** | Signals that are isolated from the bus if, and only if, the MRL is open. They consist of **PME{n}#**, **3.3Vaux{n}**, **SMBCLK{n}**, and **SMBDAT{n}**. |
| **Power Controller** | Electronic component or components that control power to a slot and monitor that power for fault conditions. |
| **primary bus** | For a PCI-to-PCI bridge that is integrated with an SHPC, this term refers to the primary PCI bus. For a host bridge that is integrated with an SHPC, this term refers to the host bus. |
| **secondary bus** | For a PCI-to-PCI bridge that is integrated with an SHPC, this term refers to the secondary PCI bus. For a host bridge that is integrated with an SHPC, this term refers to the PCI bus that is created by the bridge. |
| **Slot Control Logic** | Electronic component or components that are responsible for disconnecting the slot from the secondary bus when a power fault occurs. When serial interfaces are used, these component(s) also convert serial-format hot-plug information from the SHPC to parallel form for use by the hot-plug support electronics at the slot. |
| **slot state** | A logical state assigned to each hot-plug slot. The slot state is stored inside the SHPC. The SHPC uses this information to control the slot power and bus switches to turn the slot on and off. In power fault conditions, the logical slot state is not necessarily the same as the electrical state of the slot. See Section 3.1.2.2. |

| | |
|---|---|
| **Standard Hot-Plug Controller (SHPC)** | An implementation of a Hot-Plug Controller as defined in PCI HP 1.1 that is compliant with this specification. |
| **Standard Usage Model** | The hot-plug usage model defined in Section 2 of this specification. |
| **switched power** | Power supply voltages at a hot-plug slot that are switched off when the slot is disabled. |
| **System Interrupt** | The mechanism by which the SHPC issues a general-purpose interrupt to the operating system. Two mechanisms are allowed by this specification: assert an **INTx#** line as defined in PCI 2.2, Section 2.2.6, or send a Message Signaled Interrupt, as defined in PCI 2.2, Section 6.8. See Section 4.7.3.1. |
| **system resources** | I/O Space addresses, Memory Space addresses, bus numbers, and interrupts. System resources are managed by system software. |
| **user interface (UI)** | The system software provides a user interface that allows hot-plug slots to be monitored and controlled. See Section 2.2.6. |
| **Wakeup Context** | Wakeup Context is bridge and SHPC state information and logic required to enable and assert the Wakeup Signal and to report the reason for asserting the Wakeup Signal. See Section 6.1. |
| **Wakeup Signal** | Signal that notifies the system software to wake the system or the SHPC from a low power state. In the case of a PCI-to-PCI bridge, this is **PME#**, as defined in PCI PM 1.1. In the case of a host bridge, this is a similar signal, referred to in this specification as **WAKE#**. See Section 6.1. |

# 2. Standard Usage Model

## 2.1. Why Specify a Usage Model?

A standard usage model is beneficial to customers who buy systems with hot-plug slots because many customers utilize hardware and software from different vendors. A standard usage model allows customers to use the PCI hot-plug slots on all of their systems without having to retrain operators. Throughout the remainder of this specification, the usage model described here is referred to as the Standard Usage Model.

In order to define a programming model for the SHPC, it is necessary to make some assumptions about the interface between a human operator and a hot-plug slot. The SHPC programming model includes two indicators, one optional push button, and a sensor on the manually-operated retention latch for each supported slot.

**Implementation Note: Pitfalls of Deviating from the Standard Usage Model**

Deviating from the Standard Usage Model may create issues that would not exist otherwise, such as:

- User confusion
- More extensive hardware testing
- Functional incompatibilities with system software
- Encountering untested paths in system software

## 2.2. Elements of the Standard Usage Model

**Table 2-1: Elements of the Standard Usage Model**

| Element | Purpose | Required or Optional | For Details Refer to Section(s) |
|---|---|---|---|
| Indicators | Shows the power and attention state of the slot | Required, except as noted in Section 2.3 | 2.2.1 and 2.3 |
| Manually-operated Retention Latch (MRL) | Holds add-in cards in place | Required | 2.2.2 |
| MRL Sensor | Allows the SHPC and system software to detect the MRL being opened | Required, except as noted in Section 2.3 | 2.2.3 and 2.3 |
| Electromechanical Interlock | Prevents removal of add-in cards while slot is powered | Optional | 2.2.4 |
| Attention Button | Allows user to request hot-plug operations | Optional | 2.2.5 |
| Software User Interface | Allows user to request hot-plug operations | Required | 2.2.6 |
| Slot Numbering | Provides visual identification of slots | Required | 2.2.7 |

### 2.2.1. Indicators

The Standard Usage Model assumes that the Platform provides two indicators per slot (the Power Indicator (called "slot state indicator" in PCI HP 1.1) and the Attention Indicator). Each indicator is in one of three states: on, off, or blinking. Hot-plug system software has exclusive control of the indicator states by issuing commands to the SHPC.

The SHPC controls blink frequency, duty cycle, and phase. Blinking indicators operate at a frequency of 1 to 2 Hz and 50% (+/- 5%) duty cycle. When more than one indicator is blinking, the SHPC is required to blink them synchronously and in-phase with each other for all indicators controlled by that SHPC. Blinking indicators are not required to be synchronous and in-phase if they are controlled by different SHPCs.

**Implementation Note: Blinking Indicators**

Ideally, the indicator blink rate should be 2 Hz if either of the following is true:

- The blink clock is derived from the PCI bus clock and the bus is operating at a speed of 33 MHz, 66 MHz, 100 MHz, or 133 MHz.
- The blink clock is derived from a separate clock independent of the bus speed.

The blink rate is permitted to be as low as 1 Hz in cases where the blink clock is derived from the PCI bus clock and the bus is operating at a speed that is lower than the maximum speed allowed by PCI 2.2 or PCI-X 1.0a.

Indicators must be placed in close proximity to their associated hot-plug slot so that the association between the indicators and the hot-plug slot is clear.

Both indicators are completely under the control of system software. The SHPC never changes the state of an indicator in response to an event such as a power fault or

unexpected MRL opening unless commanded to do so by software. An exception is granted to Platforms capable of detecting stuck-on power faults. In the specific case of a stuck-on power fault, the Platform is permitted to override the SHPC and force the Power Indicator to be on (as an indication that the add-in card should not be removed). In all cases, the SHPC's internal state for the Power Indicator must match the software selected state. The handling by system software of stuck-on faults is optional and not described elsewhere in the Standard Usage Model. Therefore, the Platform vendor must ensure that this optional feature of the Standard Usage Model is addressed via other software, Platform documentation, or by other means.

#### *2.2.1.1. Attention Indicator*

The Attention Indicator is yellow or amber in color and is used to indicate that an operational problem exists or that the hot-plug slot is being identified so that a human operator can locate it easily.

**Table 2-2: Attention Indicator States**

| Indicator Appearance | Meaning |
|---|---|
| Off | Normal - Normal operation |
| On | Attention - Operational problem at this slot |
| Blinking | Locate - Slot is being identified at the user's request |

### Attention Indicator Off

When the Attention Indicator is off, it means that neither the add-in card (if one is present) nor the hot-plug slot requires attention.

### Attention Indicator On

When the Attention Indicator is on, it means an operational problem exists at the card or slot.

An operational problem is a condition that prevents continued operation of an add-in card. The operating system or other system software determines whether a specific condition prevents continued operation of an add-in card and whether lighting the Attention Indicator is appropriate. Examples of operational problems include problems related to external cabling, add-in cards, software drivers, and power faults. In general, when the Attention Indicator is on, it means that an operation was attempted and failed or that an unexpected event occurred.

The Attention Indicator is not used to report problems detected while validating the request for a hot-plug operation. Validation is a term applied to any check that system software performs to assure that the requested operation is viable, permitted, and will not cause problems. Examples of validation failures include denial of permission to perform a hot-plug operation, insufficient power budget, bus speed or bus mode mismatches, and other conditions that may be detected before an operation begins. See Section 2.4.1 for more information on validation of requests initiated by using the Attention Button.

### Attention Indicator Blinking

When the Attention Indicator is blinking, it means that system software is identifying this slot for a human operator to find. This behavior is controlled by a user (for example, from a software user interface or management tool).

#### *2.2.1.2. Power Indicator*

The Power Indicator is green in color and is used to indicate the power state of the slot.

**Table 2-3: Power Indicator States**

| Indicator Appearance | Meaning |
|---|---|
| Off | Power Off - All supply voltages (except 3.3Vaux) have been removed from the slot. Insertion or removal of add-in cards is permitted. (Note) |
| On | Power On - The slot is powered on. Insertion or removal of add-in cards is not permitted. |
| Blinking | Power Transition - The slot is in the process of powering up or down. Insertion or removal of add-in cards is not permitted. |

Note: **3.3Vaux** is removed when the MRL is open.

**Power Indicator Off**

When the Power Indicator is off, it means that main power to the slot is off and that insertion or removal of an add-in card is permitted. If the Platform provides **3.3Vaux** to hot-plug slots and the MRL is closed, the MRL Switched Signals are connected to the slot even when the Power Indicator is off. The MRL Switched Signals are disconnected when the MRL is opened.

**Power Indicator On**

When the Power Indicator is on, it means that main power to the slot is on and that insertion or removal of an add-in card is not permitted.

**Power Indicator Blinking**

When the Power Indicator is blinking, it means that the slot is powering up or powering down and that insertion or removal of an add-in card is not permitted. This blinking Power Indicator also provides visual feedback to the human operator when the Attention Button is pressed. (See Section 2.2.5.)

### 2.2.2. Manually-operated Retention Latch (MRL)

An MRL is a manually-operated retention mechanism that holds an add-in card in the slot and prevents the user from removing the card. The MRL rigidly holds the card in the slot so that cables may be attached without the risk of creating intermittent contact with the PCI bus signals. MRLs that hold down two or more add-in cards simultaneously are permitted in Platforms that do not provide MRL Sensors. (See Sections 2.2.3 and 2.3.)

### 2.2.3. MRL Sensor

The MRL Sensor is a switch, optical device, or other type of sensor that reports the position of a slot's MRL to the SHPC. The MRL Sensor reports closed when the MRL is fully closed and open at all other times (that is, fully open and intermediate positions).

If **3.3Vaux** is wired to hot-plug slots, the MRL Switched Signals must be automatically removed from the slot when the MRL Sensor indicates that the MRL is open and must be restored to the slot when the MRL Sensor indicates that MRL has closed again. See Section 3.1.2.3.

Users must always notify the operating system of their intent to remove add-in cards via a software user interface or Attention Button (if present) and wait for power to be removed before opening an MRL. The MRL Sensor allows the SHPC to monitor the position of the MRL and therefore allows the SHPC to detect unexpected openings of the MRL. When an unexpected opening of the MRL associated with a slot is detected, the SHPC changes the state of that slot to disabled and notifies system software. (See Section 4.8.) The SHPC does not autonomously change the state of either the Power Indicator or Attention Indicator. Because the operating system does not have a chance to quiesce the add-in card and unload the device driver gracefully under these circumstances, the unexpected opening of an MRL causes unpredictable results (for example, data corruption, termination of system software execution, etc.) in some cases.

Implementation of MRL Sensors is required except as described in Section 2.3.

### 2.2.4. Electromechanical Interlock

An electromechanical interlock is a mechanism for physically locking the add-in card or MRL in place until the system software and SHPC release it. Implementation of the interlock is optional. There is no mechanism in the programming interface for explicit control of electromechanical interlocks. The Standard Usage Model assumes that if electromechanical interlocks are implemented, they are controlled by the same SHPC output signal that enables main power to the slot.

### 2.2.5. Attention Button

An Attention Button is a momentary-contact push-button, located adjacent to each hot-plug slot, that is pressed by the user to initiate a hot-insertion or a hot-removal at that slot.

The Power Indicator provides visual feedback to the human operator (if the system software accepts the request initiated by the Attention Button) by blinking. Once the Power Indicator begins blinking, a 5-second abort interval exists during which a second depression of the Attention Button cancels the operation.

If an operation initiated by an Attention Button fails for any reason, it is recommended that system software present a message explaining the failure via a software user interface or add the message to a system log.

Implementation of Attention Buttons is optional for the Platform. Support for the Attention Button is required for the SHPC and system software.

### 2.2.6. Software User Interface

System software provides a user interface that allows hot-insertions and hot-removals to be initiated and that allows occupied slots to be monitored. A detailed discussion of hot-plug user interfaces is operating system specific and is therefore beyond the scope of this document.

On systems with multiple hot-plug slots, the system software must allow the user to initiate operations at each slot independent of the states of all other slots, regardless of whether those slots are controlled by the same SHPC or a different SHPC. Therefore, the user is permitted to initiate a hot-plug operation on one slot using either the software user interface or the Attention Button while a hot-plug operation on another slot is in process, regardless of which interface was used to start the first operation.

### 2.2.7. Slot Numbering

PCI HP 1.1, Section 3.1.1 requires all slots to have a Physical Slot Identifier. PCI HP 1.1, Section 4.1.8, requires system software to display the Physical Slot Identifier on software user interfaces.

A Physical Slot Identifier (as defined in PCI HP 1.1, Section 1.5) consists of an optional chassis number and the physical slot number of the hot-plug slot. System software determines the physical slot number from registers in the SHPC. The chassis number is 0 for the main chassis. The chassis number for other chassis must be a non-zero value obtained from a PCI-to-PCI bridge's Chassis Number register (see PCI Bridge 1.1, Section 13.4) or from a Platform-vendor-specific mechanism if the chassis is not accessible via a PCI-to-PCI bridge.

The Standard Usage Model also requires that each physical slot number is globally unique within a chassis and that hot-plug slots on the same bus segment are sequentially numbered in increments of 1.

## 2.3. Legacy Enclosures

This section describes exceptions to the Standard Usage Model that are permitted in order to support mechanical systems that were developed prior to the publication of this specification. These exceptions are strongly discouraged for mechanical systems developed after the publication of this specification. Operating system vendors are not required to support legacy enclosure exceptions that are described in this section.

### 2.3.1. One Indicator

Platform implementations that provide a single indicator are allowed if implemented in one of the following ways:

- No Power Indicator is implemented. The Attention Indicator is displayed by itself directly as it is controlled by the SHPC. In this case, the Power Indicator signals from the SHPC are ignored and the software user interface provided by the Platform vendor for initiating hot-plug operations must clearly tell the user when it is safe to insert or remove an add-in card, since the power state of the slot is not displayed at the slot.
- The Platform vendor provides logic circuits (external to the SHPC) to combine the Power Indicator and Attention Indicator signals provided by the SHPC to control a single indicator that displays both the Attention Indicator states shown in Table 2-2 and the Power Indicator states shown in Table 2-3. The manner in which those states are combined is not specified.

Simultaneous implementation of the Attention Button and only one indicator is discouraged, because adequate feedback of Attention Button depressions is difficult without using the Power Indicator. See Section 4.5.3 for requirements related to identifying that Attention Buttons are not provided.

### 2.3.2. No MRL Sensor

MRL Sensors are optional if all of the following are true:

- The Platform vendor provides an alternate method for removing power from a slot while cards are being inserted or removed. The method is permitted to be procedural, such as documentation that describes the proper sequence for insertion and removal. More robust methods, such as an electromechanical interlock that prevents removal or insertion while the slot is powered up, are also allowed. Hot-Plug slots must be completely powered down during insertions and removals. Systems without MRL Sensors must not place any new requirements on add-in card vendors in this regard.
- The system does not supply MRL Switched Signals to any hot-plug slot.
- The Platform vendor wires the SHPC's MRL Sensor inputs to the 'MRL closed' position. (See Table 3-4 in Section 3.3.)
- The Platform vendor assures that the 'MRL Sensor Implemented' bit in the Slot Configuration register is hardware initialized to the logic zero state (meaning that 'MRL Sensors are not implemented'). See Section 4.5.3.

## 2.4. Indicator State Transitions

The diagrams on the following pages illustrate valid state transitions of the Power and Attention Indicators from a user perspective.

### 2.4.1. Power Indicator Transitions

Hot-insertion and hot-removal operations are initiated using a software user interface or Attention Button (if available). System software issues a command to the SHPC to cause the Power Indicator to blink in response to a request to perform a hot-plug operation from a software user interface or a single depression of the optional Attention Button. The Power Indicator continues to blink until the operation has been completed or until the operation is cancelled by the user or by system software.

Operations initiated using the Attention Button are permitted to be cancelled within the first 5 seconds by pressing the Attention Button a second time. System software ignores Attention Button depressions that occur while the Power Indicator is blinking unless they occur within the first 5 seconds.

### 2.4.1.1. Hot-insertions

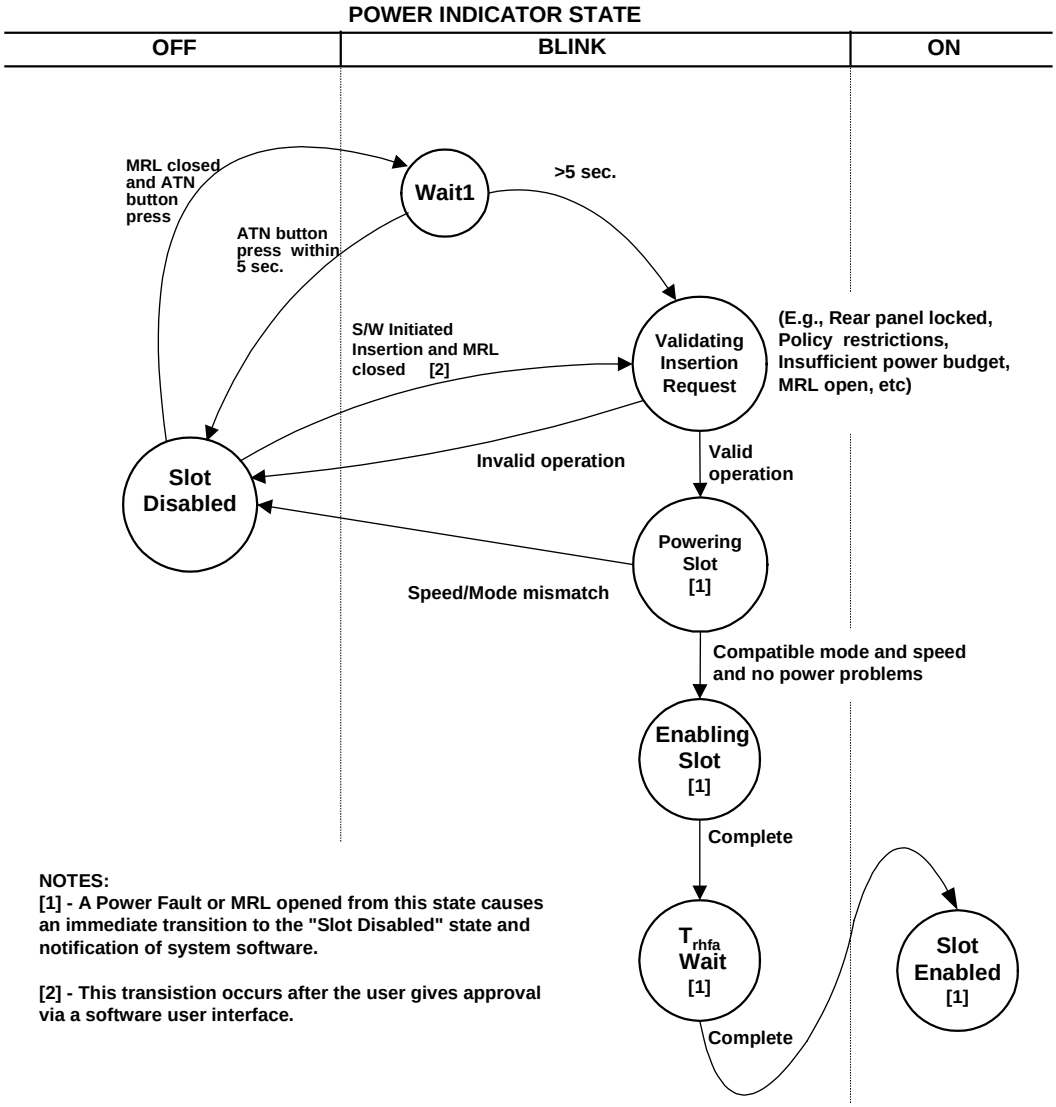



**Figure 2-1: Power Indicator State Transitions During Hot-insertion**

**Table 2-4: States and Transitions Appearing in Figure 2-1**

| State | Description and Next State (Note 1) | | Power Indicator State (Note 2) |
|---|---|---|---|
| **Slot Disabled** | Bus signals and clocks are disconnected from the slot and **RST#** is asserted to the slot. Main supply voltages are not present at the slot. If the MRL is closed, MRL Switched Signals are connected to the slot. | | Off |
| | Go to **Wait1** | If the MRL is closed and the Attention Button is pressed [SYSTEM SOFTWARE] | |
| | Go to **Validating Insertion Request** | If the MRL is closed and system software (for example, a software user interface or system management software) initiates powering or enabling the slot [SYSTEM SOFTWARE] | |
| **Wait1** | System software is waiting 5 seconds to provide the user with an opportunity to cancel the hot-plug operation. Bus signals and clocks are disconnected from the slot and **RST#** is asserted to the slot. Main supply voltages are not present at the slot. If the MRL is closed, MRL Switched Signals are connected to the slot. | | Blinking |
| | Go to **Slot Disabled** | If the Attention Button is pressed within 5 seconds [SYSTEM SOFTWARE] | |
| | Go to **Validating Insertion Request** | If 5 seconds elapses with no request to abort the operation [SYSTEM SOFTWARE] | |
| **Validating Insertion Request** | System software is verifying that the user's request to enable the slot should proceed. Bus signals and clocks are disconnected from the slot and **RST#** is asserted to the slot. Main supply voltages are not present at the slot. If the MRL is closed, the MRL Switched Signals are connected to the slot. | | Blinking |
| | Go to **Slot Disabled** | If any reason for not allowing the operation to proceed is detected by system software (the description of the Attention Indicator in Section 2.2.1.1 lists some examples of why validation may disallow the operation) [SYSTEM SOFTWARE] | |
| | Go to **Powering Slot** | If the MRL at this slot is closed and the request is valid [SYSTEM SOFTWARE] | |

| | | |
|---|---|---|
| **Powering Slot** | System software has issued the command to power the slot to determine the speed and mode capabilities of the add-in card. (The add-in card must be powered in order to check **M66EN**. See Appendix A "Presence and Capability Signals" of PCI HP 1.1.). Bus signals and clocks are disconnected from the slot and **RST#** is asserted to the slot. Main supply voltages and MRL Switched Signals are present at the slot.<br>Go to **Enabling Slot** — If no bus speed, bus mode, or power problems are detected [SYSTEM SOFTWARE]<br>Go to **Slot Disabled** — If the add-in card is not compatible with the current speed or mode of the bus or if power problems are detected [SHPC], or<br>If a Power Fault occurs at this slot [SHPC], or<br>If the MRL at this slot opens [SHPC] | Blinking |
| **Enabling Slot** | System software has issued the command to enable the slot. The SHPC connects the clock and bus signals in a specific sequence and then **RST#** is deasserted (as described in Section 3.4.3). Main supply voltages and MRL Switched Signals are present at the slot.<br>Go to **$T_{rhfa}$ Wait** — When clock and bus signals are connected to the slot and **RST#** is deasserted (at the completion of the command execution) [SYSTEM SOFTWARE]<br>Go to **Slot Disabled** — If a Power Fault occurs at this slot [SHPC], or<br>If the MRL at this slot opens [SHPC] | Blinking |
| **$T_{rhfa}$ Wait** | System software enforces the bus specification regarding the minimum time from **RST#** high to first configuration access. (See Section 4.2.3.2 of PCI 2.2.) Bus signals and clocks are connected to the slot and **RST#** is deasserted. Main supply voltages and MRL Switched Signals are present at the slot.<br>Go to **Slot Enabled** — After $T_{rhfa}$ time [SYSTEM SOFTWARE]<br>Go to **Slot Disabled** — If a Power Fault occurs at this slot [SHPC], or<br>If the MRL at this slot opens [SHPC] | Blinking |
| **Slot Enabled** | The slot is ready for operation. Bus signals and clocks are connected to the slot and **RST#** is deasserted. Main supply voltages and MRL Switched Signals are present at the slot. This is the same state as the 'Slot Enabled' state in Figure 2-2. Refer to Table 2-5 for information on transitions from this state. | On |

Notes:

1. The hardware or software entity responsible for recognizing the event and performing the transition is identified in square brackets.
2. Changes in indicator states are always under the control of system software.

### 2.4.1.2. Hot-removals

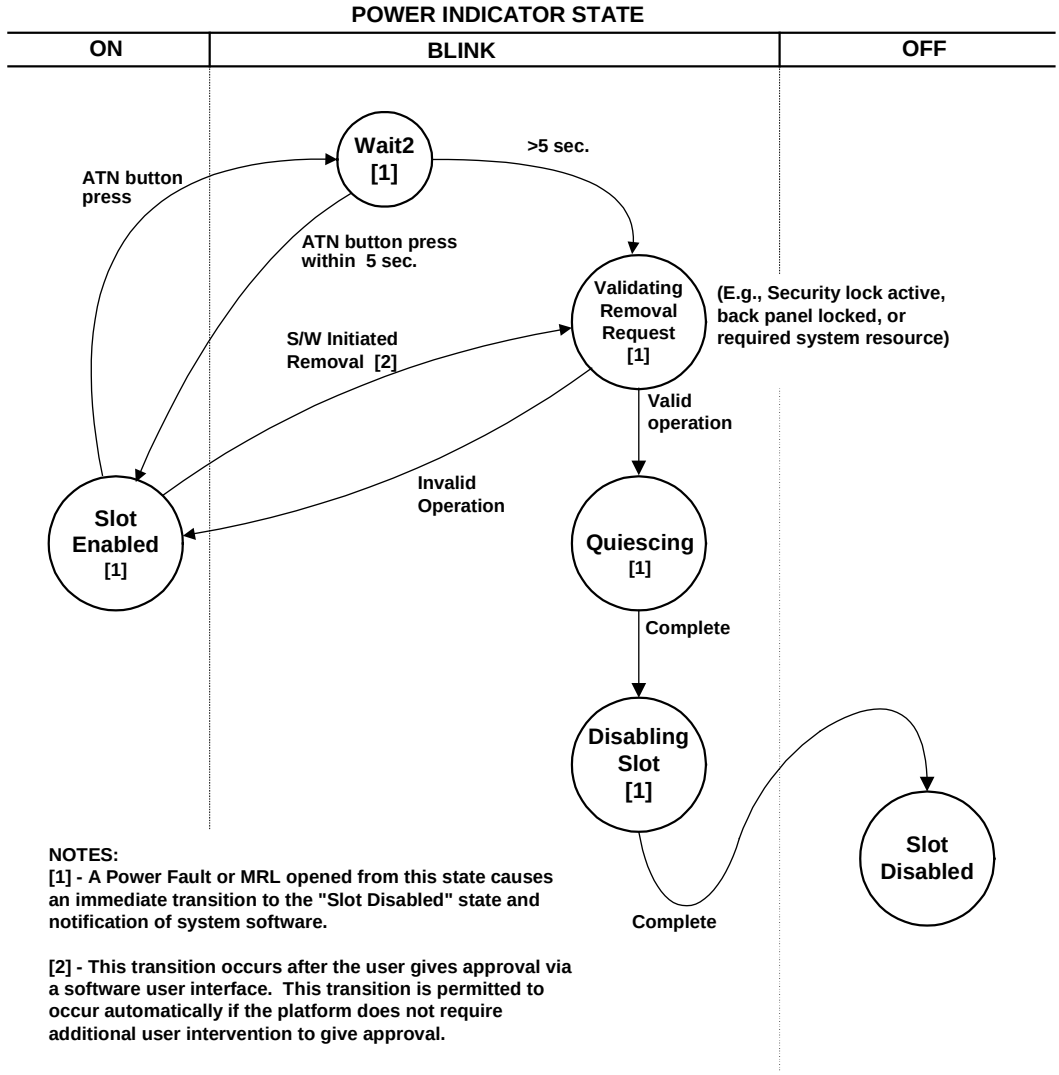



**Figure 2-2: Power Indicator State Transitions During Hot-removal**

**Table 2-5: States and Transitions Appearing in Figure 2-2**

| State | Description and Next State (Note 1) | | Power Indicator State (Note 2) |
|---|---|---|---|
| **Slot Enabled** | The slot is ready for operation. Bus signals and clocks are connected to the slot and **RST#** is de-asserted. Main supply voltages and MRL Switched Signals are present at the slot. | | On |
| | Go to **Wait2** | If the Attention Button is pressed [SYSTEM SOFTWARE] | |
| | Go to **Validating Removal Request** | If system software (for example, a software user interface or system management software) initiates disabling the slot [SYSTEM SOFTWARE] | |
| | Go to **Slot Disabled** | If a Power Fault occurs at this slot [SHPC], or<br>If the MRL opens at this slot [SHPC] | |
| **Wait2** | System software is waiting 5 seconds to provide the user with an opportunity to cancel the hot-plug operation. Bus signals and clocks are connected to the slot and **RST#** is deasserted. Main supply voltages and MRL Switched Signals are present at the slot. | | Blinking |
| | Go to **Validating Removal Request** | If 5 seconds elapses with no request to abort the operation [SYSTEM SOFTWARE] | |
| | Go to **Slot Enabled** | If the Attention Button is pressed within 5 seconds [SYSTEM SOFTWARE] | |
| | Go to **Slot Disabled** | If a Power Fault occurs at this slot [SHPC], or<br>If the MRL opens at this slot [SHPC] | |
| **Validating Removal Request** | System software is verifying that the user's request to disable the slot should proceed. Bus signals and clocks are connected to the slot and **RST#** is deasserted. Main supply voltages and MRL Switched Signals are present at the slot. | | Blinking |
| | Go to **Slot Enabled** | If any reason for not allowing the operation to proceed is detected by system software (the description of the Attention Indicator in Section 2.2.1.1 lists some examples of why validation may disallow the operation) [SYSTEM SOFTWARE] | |
| | Go to **Slot Disabled** | If a Power Fault occurs at this slot [SHPC], or<br>If the MRL opens at this slot [SHPC] | |
| | Go to **Quiescing** | If no reasons for disallowing the operation are detected [SYSTEM SOFTWARE] | |

| | | |
|---|---|---|
| **Quiescing** | System software is quiescing the device. Bus signals and clocks are connected to the slot and **RST#** is deasserted. Main supply voltages and MRL Switched Signals are present at the slot.<br>Go to **Disabling Slot** — When the device is quiesced and ready for removal [SYSTEM SOFTWARE]<br>Go to **Slot Disabled** — If a Power Fault occurs at this slot [SHPC], or<br>If the MRL opens at this slot [SHPC] | Blinking |
| **Disabling Slot** | System software has issued the 'disable slot' command and is waiting for the SHPC to assert **RST#**, disconnect bus signals and clock lines, and remove power in a specific sequence (as described in Section 3.4.4).<br>Go to **Slot Disabled** — When the add-in card is disabled (at the completion of command execution) [SYSTEM SOFTWARE], or<br>If a Power Fault occurs at this slot [SHPC], or<br>If the MRL opens at this slot [SHPC] | Blinking |
| **Slot Disabled** | Bus signals and clocks are disconnected from the slot and **RST#** is asserted to the slot. Main supply voltages are not present at the slot. If the MRL is closed, the MRL Switched Signals are applied to the slot. This is the same state as 'Slot Disabled' in Figure 2-1. Refer to Table 2-4 for information on transitions from this state. | Off |

Notes:

1. The hardware or software entity responsible for recognizing the event and performing the transition is identified in square brackets.
2. Changes in indicator states are always under the control of system software.

## 2.4.2. Attention Indicator Transitions

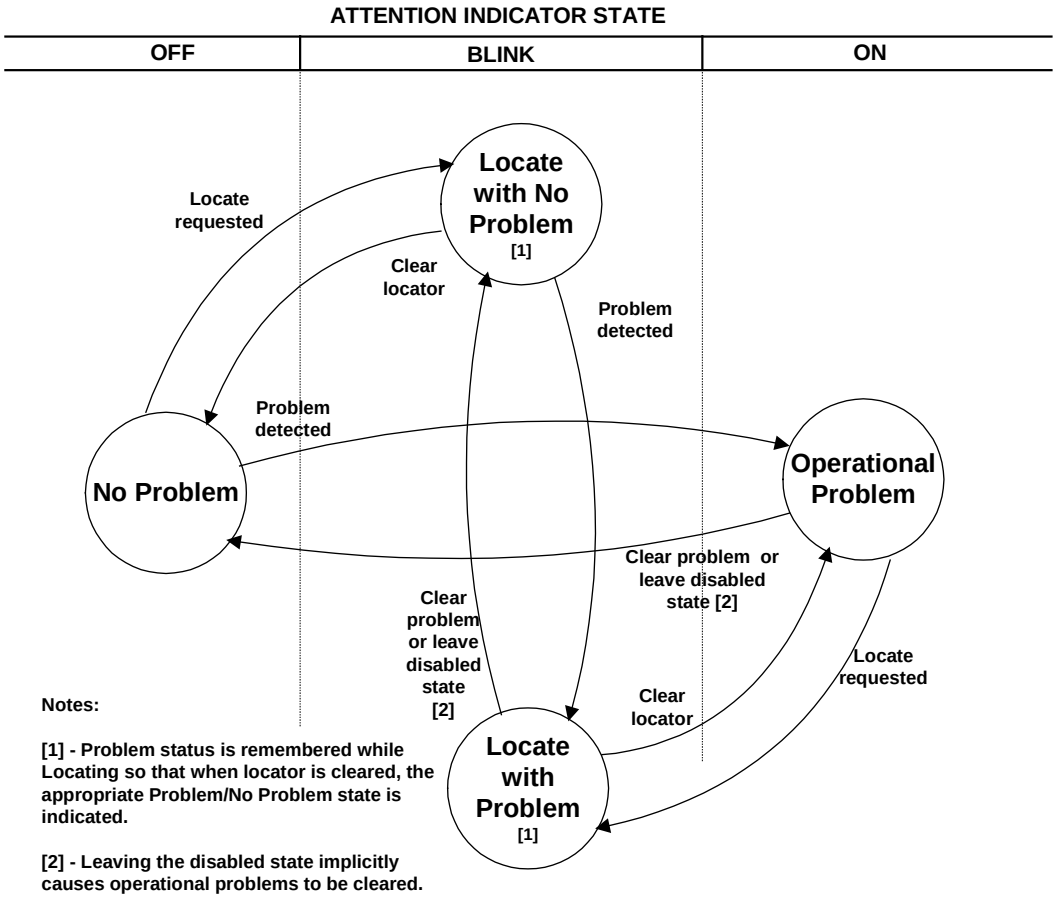



**Figure 2-3: Attention Indicator State Transitions**

**Table 2-6: States and Transitions Appearing in Figure 2-3**

| State | Description and Next State (Note 1) | | Attention Indicator State (Note 2) |
|---|---|---|---|
| **No Problem** | The Attention Indicator is not being used to locate this slot and no operational problems have been detected at the slot. | | Off |
| | Go to **Operational Problem** | If an operational problem involving this slot is detected [SYSTEM SOFTWARE] | |
| | Go to **Locate with No Problem** | If system software receives a request to blink the Attention Indicator to locate this slot [SYSTEM SOFTWARE] | |
| **Locate with No Problem** | The Attention Indicator is blinking to locate this slot and no operational problems involving this slot exists. | | Blinking |
| | Go to **No Problem** | If system software has no pending requests to locate the slot [SYSTEM SOFTWARE] | |
| | Go to **Locate with Problem** | If an operational problem involving this slot is detected [SYSTEM SOFTWARE] | |
| **Locate with Problem** | The Attention Indicator is blinking to locate this slot and one or more operational problems involving this slot exist. | | Blinking |
| | Go to **Locate with No Problem** | If all operational problems involving this slot are resolved [SYSTEM SOFTWARE], or<br>If the slot leaves the disabled state [SYSTEM SOFTWARE] | |
| | Go to **Operational Problem** | If system software has no pending requests to locate the slot [SYSTEM SOFTWARE] | |
| **Operational Problem** | The Attention Indicator is not being used to locate this slot and one or more operational problems involving this slot exist. | | On |
| | Go to **Locate with Problem** | If system software receives a request to blink the Attention Indicator to locate this slot [SYSTEM SOFTWARE] | |
| | Go to **No Problem** | If all operational problems involving this slot are resolved [SYSTEM SOFTWARE], or<br>If the slot leaves the disabled state [SYSTEM SOFTWARE] | |

Notes:

1. The hardware or software entity responsible for recognizing the event and performing the transition is identified in square brackets.
2. Changes in indicator states are always under the control of system software.

### *2.4.2.1. Special Notes Regarding the Attention Indicator*

#### 2.4.2.1.1. Locating a Slot Has Precedence Over Operational Problems

The blinking state of the Attention Indicator (to locate a slot) has precedence over other uses of the Attention Indicator. The Attention Indicator blinks when requested to do so whether there are operational problems involving the slot or not. Operational problems that develop while the Attention Indicator is blinking do not result in visible changes to the Attention Indicator until system software causes the Attention Indicator to stop blinking.

#### 2.4.2.1.2. System Software Tracks Operational Problems

Normally, the Attention Indicator returns to its previous state after being used to locate a slot. However, if an operational problem occurs while the Attention Indicator is blinking, the Attention Indicator turns on when the blinking stops to indicate that an operational problem exists. Likewise, if all operational problems are resolved while the Attention Indicator is blinking, the Attention Indicator turns off when the blinking stops to indicate that normal operation has been restored.

#### 2.4.2.1.3. Leaving the Disabled State

Whenever a hot-plug slot leaves the disabled state, operational problems associated with the slot are implicitly cleared and the Attention Indicator is turned off by system software. If an operational problem subsequently occurs or if a request is made to locate the slot, the Attention Indicator state changes appropriately.

## 2.5. User Interaction with the System

The following are examples of user interaction with the system under the Standard Usage Model. (The steps show the sequence from the perspective of the user. Some details of the interaction between hardware and software are not included.)

- Normal hot-insertion initiated via Software UI

**Table 2-7: Hot Insertion Initiated via Software UI**

| Step | User Interaction | System Response/Notes |
|---|---|---|
| 1 | User selects an empty, disabled slot and opens the MRL. (Note) | MRL Switched Signals are removed from the slot when MRL opens. |
| 2 | User inserts add-in card, closes MRL, and attaches cables to add-in card. | MRL Switched Signals are re-applied to the add-in card when the MRL closes. |
| 3 | User requests (via a software UI) that a slot be enabled. | A window or message requesting confirmation of the action appears. While waiting for the user to respond, the Power Indicator next to the slot remains off. |
| 4 | User approves the request via the software UI. | The Power Indicator next to the slot blinks while system software validates the operation. (System software is permitted to disallow a slot from being enabled. For example, it is possible that security policy settings prohibit the slot from being enabled or that the add-in card in the slot is not capable of operating at the current bus speed and mode. If the operation is cancelled by system software, the Power Indicator turns off and the slot is left powered down and disconnected from the bus. Under these circumstances, the user should check for a message or log entry that gives a specific reason for denying the request.) |
| 5 | User waits for the slot to be enabled. | When the slot has been fully enabled, the Power Indicator is turned on. |

Note: If users wish to install an add-in card into a slot that is empty but powered, they must first use the Attention Button or software UI to disable the slot as described in Table 2-9 or Table 2-10.

- Normal hot-insertion initiated via an Attention Button

**Table 2-8: Hot-insertion Initiated via Attention Button**

| Step | User Interaction | System Response/Notes |
|---|---|---|
| 1 | User selects an empty, disabled slot and opens the MRL. (Note) | MRL Switched Signals are removed from the slot when MRL opens. |
| 2 | User inserts add-in card, closes MRL, and attaches cables to add-in card. | MRL Switched Signals are re-applied to the add-in card when the MRL closes. |
| 3 | User requests that a slot be enabled by pressing the Attention Button at that slot. | The Power Indicator next to the slot blinks. (The user is permitted to cancel the request within 5 seconds by pressing the Attention Button a second time. If cancelled, the Power Indicator turns off and the slot is left powered down and disconnected from the bus.) |
| 4 | User waits 5 seconds; 'abort interval' closes. | The Power Indicator next to the slot blinks while system software validates the operation. (System software is permitted to disallow a slot from being enabled. For example, it is possible that security policy settings prohibit the slot from being enabled or that the add-in card in the slot is not capable of operating at the current bus speed and mode. If the operation is cancelled by system software, the Power Indicator turns off and the slot is left powered down and disconnected from the bus. Under these circumstances, the user should check for a message or log entry that gives a specific reason for denying the request.) |
| 5 | User waits for the slot to be enabled. | When the slot has been fully enabled, the Power Indicator is turned on. |

Note: If users wish to install an add-in card into a slot that is empty but powered, they must first use the Attention Button or software UI to disable the slot as described in Table 2-9 or Table 2-10.

- Hot-removal initiated via Software UI

**Table 2-9: Hot-removal Initiated via Software UI**

| Step | User Interaction | System Response/Notes |
|---|---|---|
| 1 | User selects a slot holding an enabled add-in card and interacts with a software UI to request that the slot be disabled. (Note) | A window or message requesting confirmation of the action appears. While waiting for the user to respond, the Power Indicator next to the slot remains on. |
| 2 | User approves the request via the software UI. | The Power Indicator next to the slot blinks while system software validates the operation. (System software is permitted to disallow a slot from being disabled. For example, it is possible that security policy settings prohibit the slot from being disabled or that the add-in card in the slot provides critical system resources. If the operation is cancelled by system software, the Power Indicator turns on and the slot is left powered up and connected to the bus. Under these circumstances, the user should check for a message or log entry that gives a specific reason for denying the request.) |
| 3 | User waits for the slot to be disabled. | When the slot has been fully disabled, the Power Indicator turns off. |
| 4 | User detaches cables, opens the MRL, and removes the add-in card. | MRL Switched Signals are removed from the slot when MRL opens. |

Note: This procedure may also be used to disable an empty, powered slot in anticipation of inserting an add-in card.

- Hot-removal using Attention Button

**Table 2-10: Hot-removal Initiated via Attention Button**

| Step | User Interaction | System Response/Notes |
|---|---|---|
| 1 | User selects a slot holding an enabled add-in card and presses the Attention Button to request that the slot be disabled. (Note) | The Power Indicator next to the slot blinks. (The user is permitted to cancel the request within 5 seconds by pressing the Attention Button again. If cancelled, the Power Indicator turns on and the slot is left powered up and connected to the bus.) |
| 2 | User waits 5 seconds; 'abort interval' closes. | The Power Indicator next to the slot continues to blink while system software validates the operation. (System software is permitted to disallow a slot from being disabled. For example, it is possible that security policy settings prohibit the slot from being disabled or that the add-in card in the slot provides critical system resources. If the operation is cancelled by system software, the Power Indicator turns on and the slot is left powered up and connected to the bus. Under these circumstances, the user should check for a message or log entry that gives a specific reason for denying the request.) |
| 3 | User waits for slot to be disabled. | When the slot has been fully disabled, the Power Indicator turns off. |
| 4 | User detaches cables, opens the MRL, and removes the add-in card. | MRL Switched Signals are removed from the slot when MRL opens. |

Note: This procedure may also be used to disable an empty, powered slot in anticipation of inserting an add-in card.

- Operational problem (isolated power fault during hot-insertion)

**Table 2-11: Isolated Power Fault During Hot-insertion**

| Step | User Interaction | System Response/Notes |
|---|---|---|
| 1 | User selects an empty, disabled slot and opens the MRL. (Note) | MRL Switched Signals are removed from the slot when MRL opens. |
| 2 | User inserts add-in card, closes MRL, and attaches cables to add-in card. | MRL Switched Signals are re-applied to the add-in card when the MRL closes. |
| 3 | User requests that the slot be enabled via a software UI or Attention Button. | If initiated by the Attention Button, the Power Indicator next to the slots blinks during the 5-second 'abort interval'. If initiated by the software UI, the Power Indicator remains off while the user is prompted for confirmation.<br><br>Once confirmed (by expiration of the 5-second 'abort interval' or explicit user input), the Power Indicator blinks while the SHPC applies power to the slot (but has not yet connected the clock and bus signals).<br><br>A power fault detected at this time is isolated from other add-in cards. When a power fault is detected, the Power Controller automatically removes power from the slot and notifies system software via the SHPC. |

Note: If users wish to install an add-in card into a slot that is empty but powered, they must first use the Attention Button or software UI to disable the slot as described in Table 2-9 or Table 2-10.

- Operational problem (connected power fault during/after hot-insertion)

**Table 2-12: Connected Power Fault During/After Hot-insertion**

| Step | User Interaction | System Response/Notes |
|---|---|---|
| 1 | User selects an empty, disabled slot and opens the MRL. (Note 1) | MRL Switched Signals are removed from the slot when MRL opens. |
| 2 | User inserts add-in card, closes MRL, and attaches cables to add-in card. | MRL Switched Signals are re-applied to the add-in card when the MRL closes. |
| 3 | User requests that the slot be enabled via a software UI or Attention Button. | If initiated by the Attention Button, the Power Indicator next to the slots blinks during the 5-second 'abort interval'. If initiated by the software UI, the Power Indicator remains off while the user is prompted for confirmation.<br><br>Once confirmed (by expiration of the 5-second 'abort interval' or explicit user input), the Power Indicator blinks while the SHPC applies power to the slot (but has not yet connected the clock and bus signals). The SHPC eventually connects the clock and bus signals, also.<br><br>A power fault detected at this time is not isolated to a single add-in card because, in some cases, the power fault causes the clock or other bus signals to glitch and the glitch propagates to other add-in cards. (Note 2)<br><br>When a connected power fault is detected, the power controller automatically removes power from the slot and notifies system software via the SHPC. If system software is unable to determine the extent of the influence that the fault had on the rest of the system or if system software cannot contain the impact of the fault, it optionally treats the power fault as a catastrophic error and forces a shutdown of the operating system to prevent further data loss or corruption. |

Notes:

1. If users wish to install an add-in card into a slot that is empty but powered, they must first use the Attention Button or software UI to disable the slot as described in Table 2-9 or Table 2-10.

2. There is no limit to the amount of time that may pass between enabling the add-in card (that is, connecting the clock and bus signals) and the occurrence of the connected power fault.

- Locating a slot

**Table 2-13: Locating a Slot**

| Step | User Interaction | System Response/Notes |
|---|---|---|
| 1 | User requests (via software UI) that the Attention Indicator blinks to identify a slot. | System software causes the SHPC to blink the Attention Indicator at all slots that meet the selection criteria. (Note) |
| 2 | User dismisses blinking Attention Indicator (via software UI) or enables the slot. | The Attention Indicator at each affected slot stops blinking and returns to a state that accurately reflects whether or not an operational problem exists at the slot. For example, slots that had no problem and that did not develop a problem while the Attention Indicator was blinking have their Attention Indicator turn off. Slots that had a problem or that developed a problem while the Attention Indicator was blinking have their Attention Indicators turn on. |

Note: The operating system is permitted to use the blinking Attention Indicator to identify either individual slots or groups of slots (for example, all empty, 64-bit, 66-MHz slots on PCI bus number 2)

- Unexpected opening of an MRL

**Table 2-14: Unexpected Opening of an MRL**

| Step | User interaction | System Response/Notes |
|---|---|---|
| 1 | User selects an enabled slot and opens the MRL without first notifying system software of an intent to disable the slot. | The SHPC detects the opening of the MRL, disconnects the slot from the bus and clock lines, asserts **RST#**, removes power, and notifies system software via System Interrupt, **PME#**, or **SERR#** (as programmed by system software).<br><br>If system software is unable to determine the extent of the influence that the unexpected MRL opening had on the rest of the system or if system software cannot contain the impact of the unexpected MRL opening, it optionally treats the incident as a catastrophic error and forces a shutdown of the operating system to prevent further data loss or corruption. See Section 4.7.3.2. |

# 2.6. The Standard Usage Model and Power Management

## 2.6.1. Relevant System and Device Power Management States

The Platform where the SHPC resides is permitted to be in a number of different power management states, which are categorized as 'Working,' 'Standby,' 'Hibernating,' and 'Off.' The following table shows how these categories relate to state designations defined in ACPI 1.0b and 2.0:

**Table 2-15: System Power Management States**

| System State | Analogous ACPI States |
|---|---|
| 'Working' | S0 |
| 'Standby' | S1, S2, or S3 with power |
| 'Hibernating' | S4 or S4BIOS |
| 'Off' | S5 |

In some implementations, the user is not able to distinguish 'Working' from 'Standby' by observing the system. The user is typically not able to distinguish 'Hibernating' from 'Off' by observing the system. (See Section 5.3 for more information on the 'Hibernating' state.)

In addition, the system's power management software places the SHPC into any of device states ***D0***, ***D1***, ***D2***, ***D3$_{hot}$***, or ***D3$_{cold}$***. (See PCI PM 1.1, Chapter 5.) In most implementations, the power management state of the SHPC is visible only to system software and is not visible to the user.

## 2.6.2. Hot-Plug Limitations Related to Power Management

The SHPC is able to change the state of a slot in response to user input, MRL opening, or power faults only when the system state is 'Working' and the SHPC is in device state ***D0***.

When the system state is 'Standby' or the SHPC is in device state ***D1***, ***D2***, or ***D3$_{hot}$***, the system and SHPC must first be awakened before the SHPC can perform any operations to a slot. That is, the system must return to the 'Working' state and the SHPC must return to device state ***D0***. In order for this to happen, system software must enable the SHPC to wake the system. The system typically wakes when the user interacts with the mouse or keyboard or when the user presses the Attention Button at a slot. With the SHPC enabled to wake the system, SHPC device states ***D1***, ***D2***, and ***D3***, system states 'Standby' and 'Working' appear to be equivalent to the user as far as hot-plug operations are concerned.

When the system state is 'Hibernating' or 'Off,' the SHPC is typically in device state ***D3$_{cold}$*** and therefore is unable to perform any operations, including waking the system. In these states, the Power and Attention Indicators associated with each hot-plug slot are off and the MRL and Attention Button do not evoke a response.

Implementations that supply **3.3Vaux** to the SHPC and therefore support wake-up from ***D3$_{cold}$*** are permitted but are not described in this specification.

Most operating systems do not permit the hardware configuration of a system to be changed while the system state is 'Hibernating.' Since the user cannot easily differentiate between 'Hibernating' and 'Off', it is possible that an add-in card is removed, inserted, or

exchanged with a different type of add-in card while the system is 'Hibernating'. In some cases, the system is not able to resume from the 'Hibernating' state if the hardware configuration has changed. See Section 5.3 for firmware requirements in this case.

The following table summarizes the usage model in each power management state:

**Table 2-16: SHPC Power Management Usage Model**

| Power Management State | User Permitted to Enable and Disable Slots While in This Power State | Power and Attention Indicators | Attention Button and MRL | User Permitted to Insert or Remove Add-in Cards at Disabled Slots While in This Power State | Connected Power Faults |
|---|---|---|---|---|---|
| System 'Working' and SHPC in ***D0*** state | Yes – inform operating system first (via UI or Attention Button) | Normal operation as defined in Section 2.2.1 | Normal operation as defined in Sections 2.2.2 through 2.2.5 | Yes | Normal handling as defined in Section 2.5 |
| System 'Standby' or SHPC in ***D1***, ***D2***, or ***D3hot*** state | Yes – inform operating system first (via UI or Attention Button - system or SHPC wakes when user interacts with the software UI or presses Attention Button) | Normal operation as defined in Section 2.2.1 | SHPC wakes system first. After the system is awake, system response to Attention Button and MRL opening appears normal. | Yes – SHPC detects add-in card removal or insertion and wakes system | SHPC wakes system, then normal handling as defined in Section 2.5 |
| System 'Hibernating' (SHPC in ***D3cold*** state) | No | Always Off | Non responsive (system does not present software UI and ignores Attention Button) | No – Firmware performs check while returning from this state and warns user | Not Applicable – Add-in cards are not powered |
| System 'Off' (SHPC in ***D3cold*** state) | No | Always Off | Non responsive (system does not present software UI and ignores Attention Button) | Yes | Not Applicable – Add-in cards are not powered |

# 3. Platform, SHPC, and Slot-side Interface Hardware Requirements and Recommendations

This section defines hardware requirements and recommendations for the Platform, SHPC, and slot-side interface. A slot-side interface is a collection of circuits that serves as a power supply and signal interface between the secondary PCI bus and the hot-plug slot, is replicated once for each hot-plug slot implemented by the Platform, and is controlled and monitored by the SHPC.

Many implementations of a slot-side interface that conform to the SHPC specification requirements are possible. Some features of the slot-side interface are required, and are so designated in this section. Additional features of the slot-side interface are recommendations.

Although an SHPC must be integrated with a host or PCI-to-PCI bridge, this section only discusses device features that are unique to supporting hot-plug slots. Bridge features that are independent of the SHPC are controlled by PCI 2.2, PCI Bridge 1.1, and PCI-X 1.0a. The SHPC signal pins, timing behavior, and electrical characteristics are described in a way to insure compliance with PCI HP 1.1.

### Implementation Note: Deviations in the SHPC and Slot Side Interface

The recommendations included in Section 3 have been selected to comply with the requirements of PCI 2.2, PCI Bridge 1.1, PCI-X 1.0a, PCI HP 1.1, the Standard Usage Model in Section 2, and the programming interface in Section 4. Other implementations that have different signal pins, timing behavior, and electrical characteristics are possible. However, deviations from the recommendations presented here must remain compliant with the specifications named above and with other sections of the SHPC specification. Furthermore, deviations from the recommendations in this section are discouraged in cost-sensitive, high-volume applications for the following reasons:

- They tend to fragment the market for the other hot-plug-slot components (as shown in Figure 3-1 and Figure 3-2), thereby missing opportunities for economies of scale.
- The nuances of different implementations tend to increase development cost for the Platform vendor.

In general, cost-sensitive, high-volume applications should not deviate from the recommended SHPC internal design and slot-side interface specifications presented here unless the alternative implementation presents clear and compelling advantages and only after considering the potential impact to hot-plug-slot components and software.

## 3.1. Hot-Plug Subsystem Architecture

As shown in Figure 3-1 and Figure 3-2, the hot-plug subsystem of a typical hot-plug Platform is comprised of the following components:

| | |
|---|---|
| **PCI-to-PCI or Host Bridge** | A bridge with an integrated SHPC. |
| **PCI Expansion Slot** | A hot-pluggable PCI slot capable of accepting a standard form-factor PCI add-in card. |
| **Slot Control Logic** | Electronic component or components that are responsible for disconnecting the slot from the secondary bus when a power fault occurs. If serial interfaces are used (as in Figure 3-1), the Slot Control Logic also converts serial-format control information from the SHPC to parallel form for use by the hot-plug support electronics at the slot. |
| **Power Controller and Power Switches** | Electronic component or components that control power to a single slot and monitor its power for a fault condition. There is one Power Controller and one set of Power Switches for each hot-plug slot. |
| **Bus Switches** | Components, typically Field Effect Transistors (FETs), that connect the secondary bus signals to the slot. There is one set of Bus Switches for each hot-plug slot. |
| **User Controls and Indicators** | The MRL Sensor, Attention Button, Power Indicator, and Attention Indicator for the slot. There is one set of User Controls and Indicators for each hot-plug slot. |

Although Figure 3-1 and Figure 3-2 show separate component blocks for each slot, any level of component integration is permitted. Implementations using either a serial interface as shown in Figure 3-1 or a parallel interface as shown in Figure 3-2 are allowed. However, detailed descriptions in the remainder of Section 3 generally assume an SHPC implementation using a serialized slot-side interface.

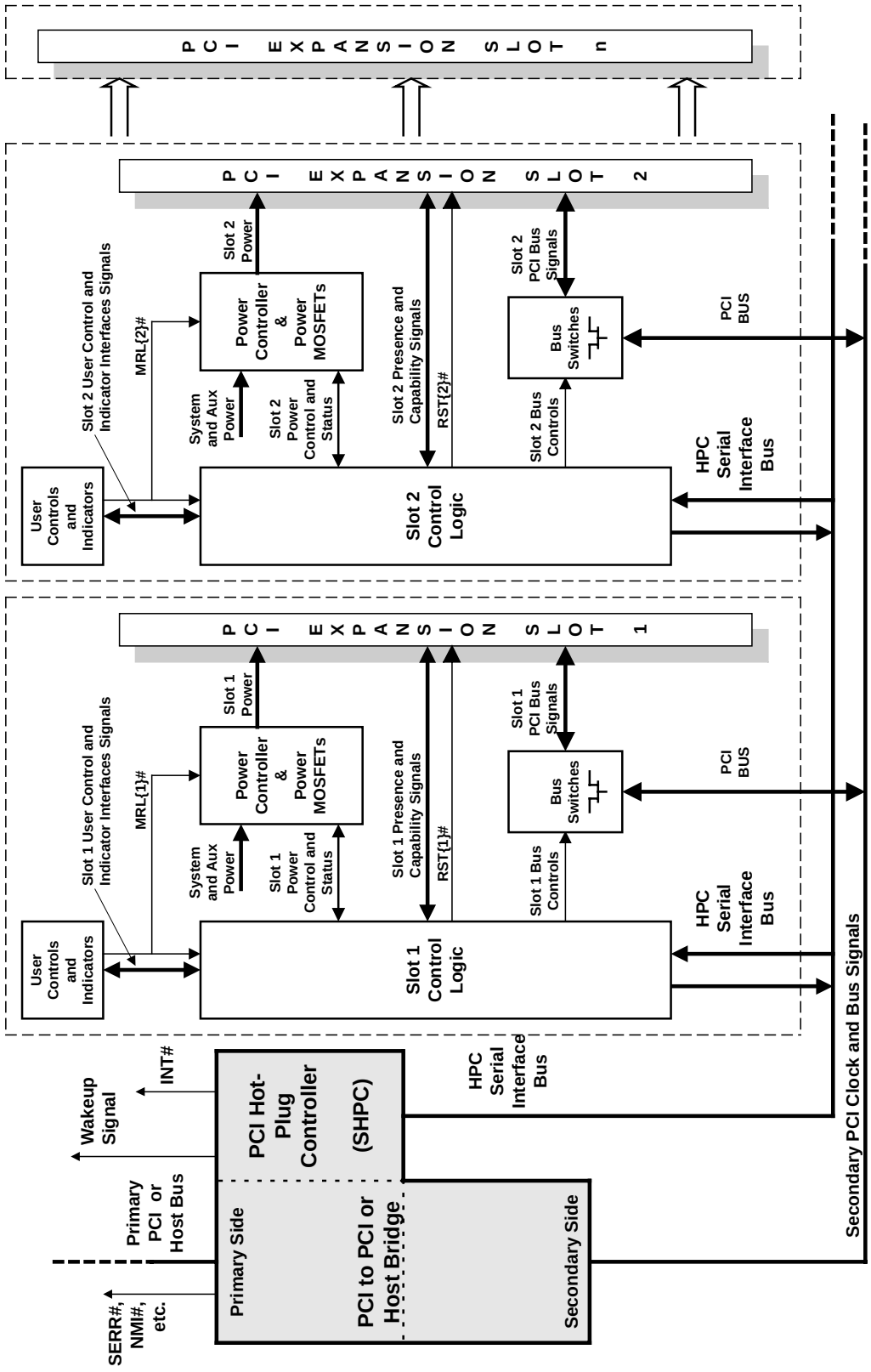



**Figure 3-1: Typical Implementation of Platform with Standard Hot-Plug Controller and Serial Interface to Slot-side Control Logic**

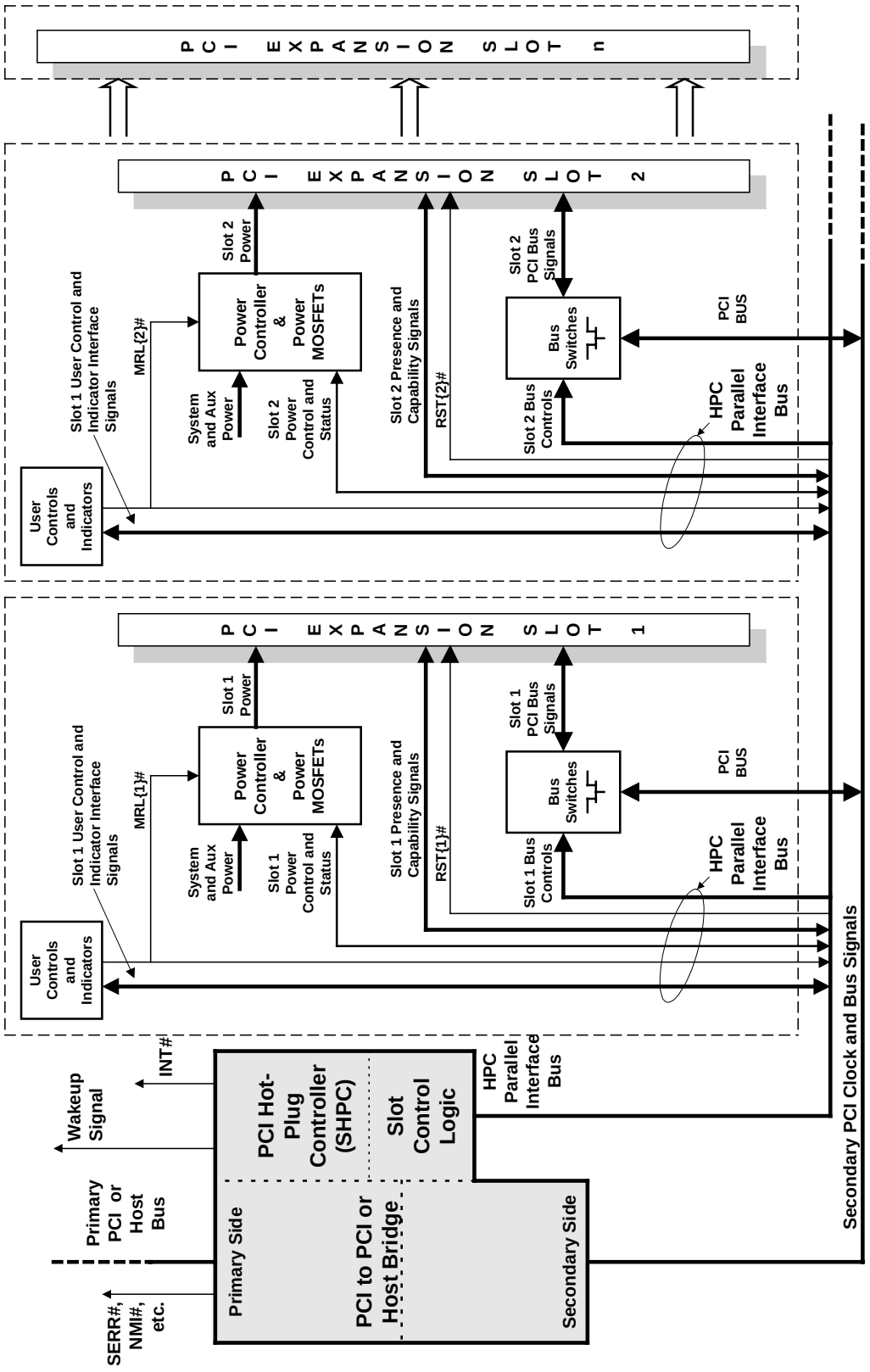



**Figure 3-2: Typical Implementation of Platform with Standard Hot-Plug Controller and Parallel Interface to Slot-side Control Logic**

### 3.1.1. Slot Control Logic

The Slot Control Logic is optionally designed to support either a serial or parallel SHPC interface. In either case, the Slot Control Logic is responsible for the following:

- Monitoring the power fault signal from the Power Controller and asynchronously asserting **RST{n}#** and disconnecting the slot from the secondary bus when a power fault occurs
- Driving the slot-specific **M66EN{n}** signal as appropriate when the slot is enabled
- Analog-to-digital level conversion for the **PCIXCAP{n}** signal

As illustrated in Figure 3-1, the SHPC uses a serial interface to communicate with external Slot Control Logic. Figure 3-2 illustrates an SHPC using a parallel slot interface whereby the Slot Control Logic is connected to the SHPC in the bridge.

If the SHPC supports a serial interface, the Slot Control Logic is also responsible for converting the serial interface signals to/from parallel form for use by the support electronics.

### Implementation Note: Serial Interfaces to Slot Control Logic

Implementations of a serial interface may take many forms. In general, simpler Slot Control Logic requires more serial interface signals; more complex Slot Control Logic reduces the number of serial interface signals.

Low speed serial interfaces, such as 400 kHz $I^2C$, are not good candidates for an SHPC serial interface. For example, in a typical slot enable sequence, the SHPC arbitrates for control of the secondary bus and drives **REQ64#** and the PCI-X initialization pattern. Then, while the SHPC retains ownership of the secondary bus, the SHPC communicates with the Slot Control Logic to deassert **RST{n}#**. The latency of communicating with the Slot Control Logic, using an $I^2C$ bus, prevents other bus masters from utilizing the bus for an extended time period, potentially causing data over/under run conditions for add-in cards.

Other design considerations for serial interfaces are:

- Protocol must support up to 31 slots. SHPCs maximally support up to 31 slots and all good serial interfaces are extensible to the maximum number of slots supported by a single SHPC.
- Protocol must support all required inputs and outputs of SHPCs (see Section 3.3).
- Protocol must support recommended inputs and outputs and be extensible to support new inputs and outputs, if any are defined in the future.
- Protocol must support simultaneous SHPC input and output. For example, while changing the state of one slot, the SHPC must be able to monitor inputs from each slot.
- Protocol must support efficient global command implementations. The Enable All Slots and Power-Only All Slots commands are required to execute within a maximum time limit specified in Section 4.6.
- Support for diagnostic, closed-loop reporting of slot-side output signal state is recommended. Though not required by the SHPC programming model, software developers, manufacturing, and service personnel welcome diagnostic features. If diagnostic commands are implemented, the Vendor-Specific commands must be used.
- Operation at clock speeds less than 20 MHz is recommended. Slot Control Logic operating at these clock speeds is easier to integrate with Power Controllers, bus switches, Power FETs, etc., (using slow, inexpensive analog processes).

## 3.1.2. Power Controller and Power Switches

### *3.1.2.1. Slot Power Enable*

The Power Controller receives a **PWREN{n}** signal from the Slot Control Logic and is responsible for powering the slot when the **PWREN{n}** signal is asserted. When the **PWREN{n}** signal is asserted, the Power Controller controls the inrush current of the slot supply rails as described in PCI HP 1.1.

### *3.1.2.2. Power Fault Detection*

The Power Controller monitors slot power for compliance with PCI 2.2. All SHPC Platforms must detect slot-specific over-current power fault conditions for each supply rail of each hot-plug slot. If the SHPC Platform supports **3.3Vaux**, over-current fault detection is also required on **3.3Vaux** of each hot-plug slot. All SHPC Platforms must detect slot-specific under-voltage conditions on the 3.3V- and 5V-supply rails of each hot-plug slot.

The required 5V, 3.3V, +12V, and –12V power rails are called "main power." The optional **3.3Vaux** supply is called "auxiliary power." The Power Controller detects and latches faults on main power independently from faults on auxiliary power. If a main power fault is detected, the Power Controller sets its internal main power fault latch and turns off main power to the slot (without affecting auxiliary power). The main power fault latch inside the Power Controller is cleared by the deassertion of the **PWREN{n}** signal. If the Power Controller supports **3.3Vaux** and an auxiliary power fault is detected, the Power Controller sets its internal auxiliary power fault latch, turns off auxiliary power to the slot (without affecting main power), and causes the **PME{n}#** and SMBus signals to be disconnected. (See Figure 3-18.) The auxiliary power fault latch inside the Power Controller is cleared when the slot's MRL is opened.

The Power Controller asserts the **PWRFLT{n}#** output if either or both internal power fault latches are set. See Section 3.1.1 for the requirement for the Slot Control Logic to assert **RST#** and disconnect the slot when the **PWRFLT{n}#** signal asserts.

### *3.1.2.3. 3.3Vaux Control*

Power Controllers designed to support **3.3Vaux** are required to monitor the MRL Sensor. The Power Controller enables slot-specific **3.3Vaux** power to the slot when the MRL is closed and disables **3.3Vaux** power when the MRL is open. When slot support for **3.3Vaux** is implemented, the Power Controller must be able to determine the state of the MRL Sensor when main power to the Platform is off. Therefore, both the MRL Sensor electronics and at least a portion of the Power Controller electronics must be powered by **3.3Vaux** power provided by the system. The Power Controller is required to debounce both open and closed states of the MRL Sensor.

If the **PME{n}#** and SMBus signals (**SMBCLK{n}** and **SMBDAT{n}**) are implemented, the Power Controller and Slot Control Logic monitor the MRL to control the connection of these slot-specific signals to the system. These signals are not connected to the system until the **3.3Vaux** voltage rail at the slot is within specification and a $T_{epme}$ delay has elapsed (see Table 3-13). Similarly, when the MRL is opened, these signals are disconnected from the system. After a delay of $T_{dpme}$, **3.3Vaux** power is removed from the slot. (See Table 3-13 and Figure 3-17.)

**Implementation Note: MRL Sensor Debouncing**

As illustrated in Figure 3-1 and Figure 3-2, either the Slot Control Logic or the SHPC is responsible for debouncing the MRL Sensor input to the SHPC. The MRL Sensor output is permitted to be debounced in one place and the debounced sensor signal distributed to all devices requiring the sensor input. Alternatively, each device requiring an MRL Sensor input could debounce the sensor signal independently. See Appendix A for additional information regarding switch debounce techniques.

### 3.1.3. PCI Bus Signals

All secondary bus signals except **RST{n}#** and the Presence and Capability Signals (listed in Table 3-3) pass through electronic switches to allow the slot to be isolated from the operational secondary bus. The electronic switches connect the secondary bus signals to the slot in a bi-directional manner. Once the electronic switches are enabled, these signals are driven by the slot or by the secondary bus interface of the bridge without special logic that steers data/current flow. Enabled electronic switches have a very low impedance (typically 5 Ω) and propagation delay (typically 250 ps). Disabled electronic switches have very high impedance so that the slot is isolated from the operational secondary bus. When the electronic switches are disabled, the slot-side of each bus switch must be pulled up (typically 10 kΩ) to the switched slot I/O signaling voltage (as controlled by the Power Controller and the slot-specific **PWREN** signal). The required pull-ups are optionally permitted to be integrated with the bus switch. However, the pull-up resistors must be either switched off when the bus is connected or weak enough such that the cumulative pull-up impedance of the bus segment (when all hot-plug slots are enabled) does not violate the minimum value defined by Section 4.3.3 of PCI 2.2.

**Implementation Note: Undervoltage-Hardened Bus Switches**

As defined by PCI 2.2, Section 4.2.1.3, PCI buses normally exhibit a significant undervoltage as signals are reflected from the ends of the transmission line. If the Platform secondary bus exhibits excessive undervoltages as a result of its non-terminated transmission line environment, the use of electronic switch devices designed to be tolerant of negative signal voltages is required. Undervoltages cause certain types of electronic FET switch devices to turn on when the slot is disconnected (that is, in the disabled state). This occurs because the voltage potential of the FET source node (connected to a PCI bus signal) is more negative than the FET gate node (connected to **BUSEN#**) by a value greater than the FET gate threshold.

### 3.1.4. PCI RST#

The slot-specific **RST{n}#** pin is driven directly by the Slot Control Logic. The electrical requirements for this signal are defined by PCI 2.2.

PCI Bridge 1.1, Section 11.1.1 requires secondary **RST#** to be asserted asynchronously with respect to the secondary bus CLK and permits it to be deasserted either asynchronously or synchronously. The SHPC must asynchronously assert all slot-specific **RST{n}#** whenever secondary **RST#** asserts. When secondary bus **RST#** deasserts, the SHPC must deassert **RST{n}#** as described in Section 3.4.5 in a manner compliant with PCI 2.2. One or more slot-specific **RST{n}#** signals also assert and deassert in other cases described in Section 3.4.5.

If the Power Controller for the slot detects a power fault, the Slot Control Logic asynchronously asserts **RST{n}#** to the slot (independent of the SHPC) to isolate the faulty slot in the most expedient manner possible. See PCI 2.2, Section 4.3.2, for the specification for the delay from a power fault to the assertion of **RST{n}#**.

### 3.1.5. Presence and Capability Signals

PCI signals that are used to communicate the presence of an add-in card and its capability consists of **PRSNT1{n}#**, **PRSNT2{n}#**, **M66EN{n}**, and **PCIXCAP{n}**. Although a typical non-hot-plug system wire-ORs the **M66EN** and **PCIXCAP** pins of each slot, in

hot-plug systems these signals are kept isolated from other slots. For additional information, refer to PCI HP 1.1.

The **PRSNT1{n}#**, **PRSNT2{n}#**, and **PCIXCAP{n}** pins are inputs to the Slot Control Logic. However, since **M66EN** is defined by PCI 2.2 to be a bi-directional signal, special consideration is required. The recommended slot-side interface circuit implementation is discussed here.

To determine the 66 MHz conventional mode capability of an add-in card in a disabled slot, the system software instructs the SHPC to apply power to the slot without connecting it to the bus. After the 66 MHz conventional mode capability of an add-in card has been determined, software instructs the SHPC to enable the slot, thereby connecting it to the bus. During the slot enable sequence, the SHPC and Slot Control Logic are responsible for driving the **M66EN{n}** input low if the bus is not operating in 66 MHz conventional mode. The implementation presented in Figure 3-3 uses the slot-specific control signal **CLKEN{n}#** to connect a global (bus-segment specific) frequency status signal (**M66OUT**) to the slot's **M66EN{n}** pin as the slot's clock is connected. The global **M66OUT** signal is high when the bus is operating in conventional PCI mode at 66 MHz. The value of **M66EN{n}** that is sampled before it is connected to the bus is reported via the 66 MHz Capable bit in the Slot Status field of the Logical Slot register (see Section 4.5.12.1). Component values shown in Figure 3-3 are typical.

In the timing diagram at the bottom of Figure 3-3, the solid line on **M66EN{n}** illustrates the case in which the card is capable of 66 MHz operation (**M66EN{n}** goes high when power is applied), but the bus is presently operating in 33 MHz conventional mode (**M66OUT** is low).

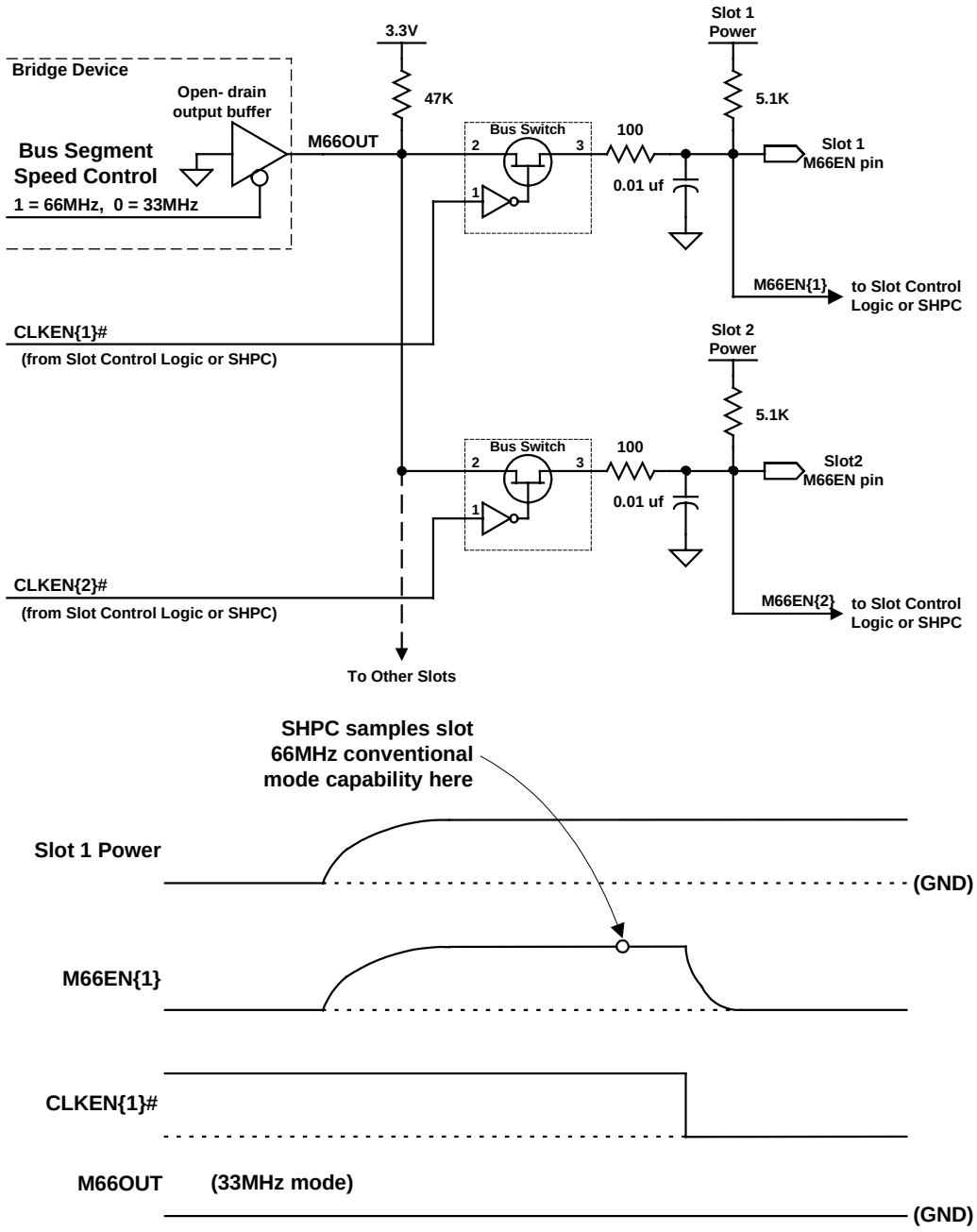



**Figure 3-3: M66EN Interface Circuit**

### 3.1.6. User Controls and Indicators

The SHPC controls and monitors slot-specific user-interface devices consisting of the MRL{n} Sensor, the Attention Button, the Power Indicator, and the Attention Indicator. It is recommended that the indicator outputs be open drain and high-current. The output currents specified in Table 3-12 were chosen such that two indicators (LED's) can be driven directly from the Slot Control Logic. (Multiple indicators are often required by Platform designers so that they are visible from both the front and rear of the Platform.)

If the Platform does not support an MRL Sensor, the **MRL{n}#** input to the Slot Control Logic is wired to a low logic voltage and the hardware-initialized MRL Sensor Implemented bit in the Slot Configuration register is cleared by the Platform (see Section 4.5.3).

If the Platform does not support an Attention Button, the **BUTTON{n}#** input to the Slot Control Logic is connected to a high logic voltage, and the hardware-initialized Attention Button Implemented bit in the Slot Configuration register is cleared by the Platform (see Section 4.5.3).

### 3.1.7. System Interrupt, Wakeup Signal, and System Error

A bridge with an integrated SHPC is required to support a System Interrupt. The SHPC programming model uses the System Interrupt to interrupt the operating system and notify it that the SHPC needs attention. See Section 4.7.3.1 for additional information about System Interrupt.

The **PME#** and **WAKE#** pins are used to wake the Platform when the SHPC is in the ***D1***, ***D2***, or ***D3*** power-managed state and when the system is in the 'Standby' state (see Section 2.6.2). A PCI-to-PCI bridge with an integrated SHPC is required to support the **PME#** pin. If a host bridge with an integrated SHPC is intended for a system that supports the 'Standby' state, that host bridge is required to support the **WAKE#** pin. If a host bridge with an integrated SHPC is intended for a system that does not support the 'Standby' state, the **WAKE#** pin is optional.

PCI-to-PCI bridges must implement an **SERR#** pin (as required by PCI Bridge 1.1). Host bridges must implement a means to generate a critical system interrupt, for example, NMI or machine check.

## 3.2. Implementation Choices

The SHPC specification allows considerable latitude with regard to compliant SHPC and Platform implementations. To the extent that many of the features described in this specification are optional or recommended, this section discusses features that potentially differentiate the SHPC subsystems of Platform vendors.

### 3.2.1. Number of Slots

The SHPC and its programming model optionally supports from one to thirty-one hot-plug slots. The number of slots to be supported by a particular implementation of the SHPC is a function of many factors such as market requirements, bus loading limitations, pin count constraints, etc.

### 3.2.2. Serial or Parallel Slot Interface

A large number of signals are needed to support each hot-plug slot. Most SHPCs are physically integrated with host or PCI-to-PCI bridges. In many cases, the pin count impact of supporting hot-plug slots significantly increases the cost of the bridge component. Consequently, Platform designers designing cost-effective systems will consider using a low-pin-count serial interface to communicate with slot-specific hot-plug infrastructure (bus switches, user switches, user buttons, power controllers, etc.), as shown in Figure 3-1. This specification does not define a specific implementation of such a serial interface. However, guidelines for developers are discussed in Section 3.1.1.

### 3.2.3. Power-Managed SHPC and/or Slots

The following features are optional for the SHPC:

- **3.3Vaux** supply – As defined in PCI PM 1.1, **3.3Vaux** is used to allow hot-plug operations while the bridge/SHPC is in a power-managed state in which main power is removed. The decision to provide **3.3Vaux** support for hot-plug slots is independent of whether **3.3Vaux** is provided to the bridge/SHPC.
- SMBus support.
- **WAKE#** signal (for a host bridge).

The following features are optional for the hot-plug slot:

- **3.3Vaux** supply
- **PME#** support
- SMBus support (See SMBus ECN.)

### 3.2.4. Capability Signals and REQ64#

If the secondary interface of the bridge integrated with an SHPC does not support PCI-X mode, the **PCIXCAP** input from the hot-plug slot is not required. In this case, the PCI-X Capability bits in the Slot Status field of the Logical Slot register are implemented as read-only and return a value of 00b (non-PCI-X) and the fields of the Slots Available registers must indicate no PCI-X slots are available. See Sections 4.5.12.1 and 4.5.2.

If the secondary interface of the bridge integrated with an SHPC does not support 66 MHz conventional mode, the **M66EN** input from the hot-plug slot is not required. In this case, the 66 MHz Capable bit in the Slot Status field of the Logical Slot register is implemented as read-only and returns a value of 0 and the fields of the Slots Available registers must indicate no 66 MHz conventional-mode slots are available. See Sections 4.5.12.1 and 4.5.2.

If the bus segment controlled by the SHPC does not support 64-bit slots, the **REQ64#** pin of the slot (**REQ64{n}#**) must be pulled up to slot power.

### 3.2.5. MRL's, Attention Buttons, and Indicators

Platform designers optionally implement systems with any combination of the following:

- One indicator per slot (see Section 2.3.1)
- No MRL Sensor (see Sections 2.3.2 and 4.5.3)
- No Attention Button (see Sections 2.2.5 and 4.5.3)

### 3.2.6. Clock Sources

The bridge component must operate in a minimum of two clock domains, that is, the primary/host bus clock and the secondary bus clock. The frequencies of these two clocks are not necessarily the same. A bridge with an integrated SHPC must support the ability to change the bus frequency and mode of the secondary bus. From the perspective of the SHPC core logic, some aspects of the SHPC must be accessible while the secondary bus is being reset (for example, Command Completion Detected register), and designers are recommended to implement these circuits within the primary or host bus clock domain. Other aspects of the SHPC must interface with the secondary bus (for example, circuits interfacing to **REQ64#**, **PHP_REQ#**, and **PHP_GNT#**), and designers must implement these circuits within the secondary bus clock domain. Consequently, an SHPC is designed to operate from some combination of the following clock sources:

**Primary Bus Clock** — The programming model of the SHPC is accessed through the primary interface of the bridge. SHPC registers are clocked by the primary bus clock.

**Secondary Bus Clock** — Some aspects of hot-plug control require interface to the secondary bus. For example, when an SHPC enables a slot, it must drive a terminating pattern on the secondary bus and must conform to the timing requirements imposed by PCI 2.2, Section 4.3.2.

**Independent (Continuously-Available) Clock**

As described in Section 6.2.1, a PCI-to-PCI bridge that asserts **PME#** from ***D2*** and $D3_{hot}$ can optionally be designed to require the primary clock to remain operational. Host bridges are allowed the equivalent option, but the details of the implementation are not specified. If a bridge design does not require the primary clock to remain operational, an independent clock source is used for slot-side interface purposes. In such implementations, the SHPC uses the independent clock domain to monitor user controls and update indicators while the SHPC is in ***D2*** or $D3_{hot}$ power-managed state. The PME Clock bit (in the PCI Power Management Capabilities List Item) or equivalent host bridge control bit is cleared to indicate the SHPC is capable of generating the Wakeup Signal when the primary-side PCI clock is stopped.

### 3.2.7. Non-Hot-Plug Devices on a Hot-Plug Bus Segment

Platform designers may optionally mix hot-plug slots and non-hot-plug devices (not in slots) on the same bus segment. Platform designers are discouraged from mixing hot-plug slots and non-hot-plug slots on the same bus segment. Mixing of hot-plug and non-hot-plug devices on the same bus segment is allowed if the initialization of all Number of Slots Available fields in the Slots Available I and Slots Available II registers takes into account the capabilities and loading of all devices and slots on the bus segment as described in Section 4.5.2. The implementation note below discusses considerations and mechanisms for adjusting the contents of the Number of Slots Available fields in the case of non-hot-plug slots.

When hot-plug and non-hot-plug devices share a common bus segment:

1. The Number of Slots Implemented field in the Slot Configuration register must be initialized to a value that matches the total number of hot-plug slots implemented on the bus segment (independent of speed or mode). The value in this field does not include non-hot-plug slots or devices.
2. All Number of Slots Available fields in the Slots Available I and Slots Available II registers must be initialized to a value that matches the maximum number of hot-plug slots that are permitted to be enabled on the bus at the same time as all of the non-hot-plug devices.
3. Devices on this bus segment must be numbered so that non-hot-plug slots are assigned device numbers lower than the value of the First Device Number field in the Slot Configuration Register and non-hot-plug devices (not in slots) are assigned device numbers greater that the last hot-plug slot.

See Section 4.5.2 for requirements for assigning device numbers to slots if the number of slots available at a particular frequency and mode is less than the number of slots implemented.

### Implementation Note: Mixing Hot-plug and Non-Hot-plug Slots on the Same Hot-plug Bus Segment

The presence of non-hot-plug slots on a hot-plug bus segment poses a unique problem with regard to initializing the Slots Available I and Slots Available II registers. At system boot or after system hibernation, the complement of add-in cards present in non-hot-plug slots may have changed since the last time the system was operating. In such cases, the information contained in the Slots Available I and Slots Available II registers may need to be changed to prevent the SHPC from initializing or placing the bus in an illegal bus speed or mode.

The SHPC does not support the collection of slot capability data from non-hot-plug slots. Therefore, the **M66EN** and **PCIXCAP** pins of the non-hot-plug slots are separately wire-ORed as is normally the case for non-hot-plug bus segments. If the SHPC is initialized by hardware (for example, a serial EEPROM), the wire-ORed capability data can be used to select the output data from two or more serial EEPROMs whose Slots Available I and Slots Available II register locations have been pre-programmed with the correct information for the non-hot-plug add-in cards present. If the SHPC is initialized by firmware, the Platform can be designed to support a mechanism for firmware to read the wire-ORed capability data from the non-hot-plug slots. After reading the capability data, the firmware determines the appropriate values for the Slots Available I and Slots Available II registers and initializes the SHPC accordingly.

## 3.3. Hot-Plug Controller Signals

The Slot Control Logic I/O signals described in Table 3-1 through Table 3-4 are unique for each supported slot, that is, slot-specific. The I/O table heading indicates whether the signal is an input or output (or both) from the perspective of the Slot Control Logic. All SHPC I/O signals are required except the **CLKEN{n}#** signal. See Section 3.2.4 for cases for which **PCIXCAP** and **M66EN** are not required.

**Table 3-1: Power Controller Interfaces**

| Signal Name | I/O | Definition |
|---|---|---|
| **PWREN{n}** | O | Power Enable Output:<br>A Slot Control Logic output that controls the power state of the slot. If this signal is high, power is enabled to the slot. |
| **PWRFLT{n}#** | I | Power Fault Input:<br>A low-true input that indicates that the power controller for the slot has detected a power fault on one or more supply rails. See Sections 3.1.2.2 and 3.1.2.3 for additional requirements. |

**Table 3-2: PCI Bus Signal Interfaces**

| Signal Name | I/O | Definition |
|---|---|---|
| **CLKEN{n}#** | O | Clock Enable Output:<br>An optional Slot Control Logic output that controls the connection of PCI clock to the slot. If this signal is low, **CLK** is enabled to the slot. When this signal is not available or used, **BUSEN{n}#** is used to enable **CLK**. (See Implementation Note: Inclusion of **CLKEN{n}#** Support.) |
| **BUSEN{n}#** | O | Bus Enable Output:<br>A Slot Control Logic output that controls the connection of all bus signals to the slot except:<br>Power rail pins, **CLK{n}** (if **CLKEN{n}#** is implemented), **RST{n}#**, **PRSNT2{n}#**, **PRSNT1{n}#**, **M66EN{n}**, **PCIXCAP{n}**, **PME{n}#**, **SMBCLK{n}**, and **SMBDAT{n}** signals. When the **BUSEN{n}#** signal is low, the secondary bus signals are enabled to the slot. |
| **RST{n}#** | O | PCI Reset Output:<br>A Slot Control Logic output which is directly connected to the slot and serves as a slot-specific **RST{n}#**. See PCI 2.2 for the specification of this signal. |

**Table 3-3: Presence and Capability Signal Interfaces**

| Signal Name | I/O | Definition |
|---|---|---|
| **PRSNT2{n}#** | I | **PRSNT2{n}#** Input:<br>A Slot Control Logic input that is directly connected to the **PRSNT2{n}#** pin of the slot. See PCI 2.2 for the specification of this signal. See PCI HP 1.1 for additional hot-plug Platform requirements. |
| **PRSNT1{n}#** | I | **PRSNT1{n}#** Input:<br>A Slot Control Logic input that is directly connected to the **PRSNT1{n}#** pin of the slot. See PCI 2.2 for the specification of this signal. See PCI HP 1.1 for additional hot-plug Platform requirements. |
| **M66EN{n**} | I/O | **M66EN** Input and Output:<br>A Slot Control Logic input that is directly connected to the **M66EN{n}** pin of the slot. See PCI 2.2 for additional pin requirements. See PCI HP 1.1 and Section 3.1.5 for additional hot-plug Platform requirements. |
| **PCIXCAP{n}** | I | **PCIXCAP** Input:<br>A control logic input that indicates whether or not the add-in card is capable of supporting PCI-X mode and, if so, its maximum operating speed. See Section 4.5.12.1 and PCI-X 1.0a for additional requirements. |

**Table 3-4: User Control and Indicator Interfaces**

| Signal Name | I/O | Definition |
|---|---|---|
| **PWRLED{n}#** | O | Power LED Output:<br>A Slot Control Logic output that is used to drive the Power Indicator. This output is asserted to illuminate the Indicator and is recommended to be open drain. |
| **ATNLED{n}#** | O | Attention LED Output:<br>A Slot Control Logic output that is used to drive the Attention Indicator. This output is asserted to illuminate the Indicator and is recommended to be open drain. |
| **MRL{n}#** | I | Manually-operated Retention Latch Sensor Input:<br>A low-true (low logic level when the MRL is closed) Slot Control Logic and Power Controller input that is connected directly to the MRL Sensor. If the Platform does not support MRL Sensors, this input must be wired to a low logic level. |
| **BUTTON{n}#** | I | Attention Button Input:<br>A low-true (low logic level when the button is pressed) Slot Control Logic input that is connected directly to the Attention Button. If the Platform does not support Attention Buttons, this input must be wired to a high logic level. |

### Implementation Note: Inclusion of CLKEN{n}# Support

Including support for the optional **CLKEN{n}#** signal in the Slot Control Logic allows the connection of the **CLK{n}** and **M66EN{n}** to occur simultaneously and in advance of the connection of other bus signals. Add-in cards that support 66 MHz conventional mode and that use **M66EN** as an input are provided a greater setup time for their Phased-Locked-Loops (PLLs) to achieve lock and for synchronous circuits to stabilize before the bus is connected. Add-in card circuitry used to monitor **M66EN** must be operational during **RST{n}#** and could be adversely affected by bus activity that occurs during the PLL lock process. When **CLKEN{n}#** support is provided, the add-in card experiences an operational environment that more closely matches that of a non-hot-plug system.

## 3.4. Power Sequencing

This section defines the power sequence timing recommendations for an SHPC and Platform. Section 3.4.3 describes the sequence used to enable a slot. Section 3.4.4 describes the sequence used to disable a slot (that is, turn off a slot that has been enabled). The timing characteristics for each sequence are divided into two parts. The first part defines an overview of the timing, wherein all phases of the enable/disable sequence are viewed in entirety. The second part defines in detail the timing, bus arbitration, and precise nature of signal switching at the bus level.

SHPC commands that enable slots and disable slots are executed in multiple phases. In each phase, the SHPC asserts or deasserts one or more output signals to control one aspect of the slot's state. When enabling a slot, the power must be turned on and the supply rails must be within specifications before the bus signals are connected to the secondary bus. In addition, PCI 2.2 and PCI-X 1.0a require some bus signals to be driven prior to the deassertion of **RST#.** Hence, multiple SHPC output controls must change state in a prescribed sequence for the slot to be added or removed from the secondary bus such that:

1. Secondary bus signal integrity remains within specification for both existing and new add-in cards
2. Platform power supply rails remain within specification

Other commands are executed in a single phase. That is, only one slot control output from the SHPC changes in response to these commands. The single-phase commands are described in Section 3.4.5.

The timing specifications included in this chapter apply to individual slots. The programming model described in Section 4.6.1 includes commands to enable or power-only all hot-plug slots with a single command. When the SHPC executes an "all slots" command, the SHPC is permitted to power up or down slots in one or more groups to minimize current transients from the power supply. Allowable implementations include the following:

- All slots change state simultaneously
- Slots change state one at a time (not in parallel)
- Slots change state in some interleaved, staggered fashion (not in parallel)

### 3.4.1. Arbitration and Bus Stabilization

Bus arbitration and stabilization is recommended during the SHPC execution of commands that result in the assertion or deassertion of **CLKEN{n}#** or **BUSEN{n}#.** Bus arbitration and stabilization is required when the SHPC deasserts **RST{n}#**. The SHPC arbitrates for control of the bus in order to stabilize the secondary bus. For the purposes of this discussion, the signals **PHP_REQ#** and **PHP_GNT#** are used to interface to the secondary bus arbiter. In an actual implementation, these signals are permitted to be either package pins or internal connections.

When stabilizing the bus, it is recommended that the SHPC drive critical secondary-bus signals to a low-impedance, deasserted state. This is done to prevent glitches caused by a bus-switch event from occurring on the bus and potentially disrupting bus transactions to/from other enabled slots. Furthermore, when a slot is being brought out of the reset state (deassertion of **RST{n}#**), the SHPC drives appropriate initialization patterns as described in PCI 2.2 and PCI-X 1.0a to initialize the slot in a mode compatible with that of the secondary bus. The SHPC drives the following signals while stabilizing:

- **AD[31::00]**
- **C/BE[3::0]#**
- **PAR**
- **FRAME#**
- **IRDY#**
- **DEVSEL#**

- **TRDY#**
- **STOP#**
- **REQ64#** (for 64-bit bus segments only)

When the bridge drives these PCI bus signals, it is required to meet the $T_{on}$, $T_{off}$, $T_{val}$, $T_{rrsu}$, and $T_{rrh}$ timing requirements defined by PCI 2.2, Section 4.2.3.2.

To initiate stabilization, the SHPC asserts **PHP_REQ#**. PCI-X 1.0a requires the arbiter to coordinate with the SHPC to keep the bus in an idle state while it drives the PCI-X initialization pattern before the rising edge of **RST{n}#**. The simplest implementation that meets this requirement is for the arbiter to treat all stabilization requests from the SHPC in the same manner. If the SHPC asserts **PHP_REQ#** to indicate its need to stabilize the bus, the arbiter asserts **PHP_GNT#** as determined by the arbitration algorithm, and it deasserts **PHP_GNT#** only after the SHPC deasserts **PHP_REQ#**. Although PCI 2.2 does not include a similar requirement for conventional-mode hot-plug controllers, the arbiter for a hot-plug subsystem using an SHPC is required to behave in this manner both in conventional and PCI-X mode.

After the assertion of **PHP_REQ#**, the SHPC starts a timer that expires within $T_{alto}$ (arbitration latency timeout). If within this time period either **PHP_GNT#** has not been asserted or the bus does not become idle after **PHP_GNT#** has been asserted, the SHPC performs the operation without obtaining an idle bus grant. If the Arbiter Timeout SERR Mask bit (located in the Slot SERR-INT Mask field of the Logical Slot register, Section 4.5.12), is cleared, the SHPC asserts system error (if enabled) for this occurrence, because there is a risk that transactions executing on the bus were affected in unpredictable ways (see Section 4.7.3.2). The time period from **PHP_REQ#** asserted to **PHP_GNT#** asserted with idle bus is defined as $T_{al}$ (arbitration latency).

### Implementation Note: External Arbiters

If the SHPC is not physically integrated with the bridge or if the secondary bus arbiter is external to the bridge, it may be advantageous to define the **PHP_GNT#** arbiter signal in such a way that it is only asserted after the bus becomes idle. This implementation avoids the need for SHPC pin connections to the **FRAME#** and **IRDY#** secondary bus signals.

### 3.4.2. Isolated and Connected Power Faults

Power faults come in two classes: isolated and connected. The slot power fault class state changes from isolated to connected when the **CLKEN{n}#** signal asserts and from connected to isolated when the **CLKEN{n}#** signal deasserts, as illustrated in Figure 3-4 and Figure 3-9, respectively. For implementations of the SHPC that do not support the **CLKEN{n}#** signal, the slot power fault class state changes when the **BUSEN{n}#** signal changes. Power faults that occur when the slot is isolated are reported as an Isolated Power Fault Detected condition in the Slot Event Latch field of the Logical Slot register. (See Section 4.5.12.) Power faults that occur when the slot is connected are reported as a Connected Power Fault Detected condition in the Slot Event Latch field.

The Slot SERR-INT Mask field of the Logical Slot register (see Section 4.5.12) is used to enable or mask power fault events from generating System Interrupts and system errors. Isolated power fault events are considered non-catastrophic and generate a System Interrupt if the Isolated Power Fault Interrupt Mask bit is cleared. Connected power fault events are potentially catastrophic and are capable of generating either a System Interrupt, a system error, or both, depending on the states of the Connected Power Fault Interrupt Mask and Connected Power Fault SERR Mask bits. Additional discussion on isolated and connected power faults can be found in Section 2.5 and Section 6.3.2.2.

### 3.4.3. Slot Enable Sequencing

When an SHPC Slot Enable or the Enable All Slots command is issued, the **PWREN{n}**, **CLKEN{n}#**, **BUSEN{n}#,** and **RST{n}#** signals for the slot(s) are switched in a prescribed sequence. When the Slot Operation command also includes Power Indicator or Attention Indicator state changes, the **PWRLED{n}#** and **ATNLED{n}#** signals are changed in accordance with the SHPC command. The SHPC executes the Slot Enable command in four phases, shown in Figure 3-4 and defined in Table 3-5. Table 3-6 defines the recommended timing between these phases.

Although the Standard Usage Model presented in Section 2.2.1.2 requires a specific relationship between the Power Indicator and the state of the slot, this relationship is enforced by software, not by the SHPC. The SHPC switches the LED outputs according to the Slot Operation command written to the Controller Command register, independent of whether that command conforms to the Standard Usage Model or not. The figures in this section illustrate the general assertion or deassertion of the LED outputs, without regard to the requirements of the Standard Usage Model.

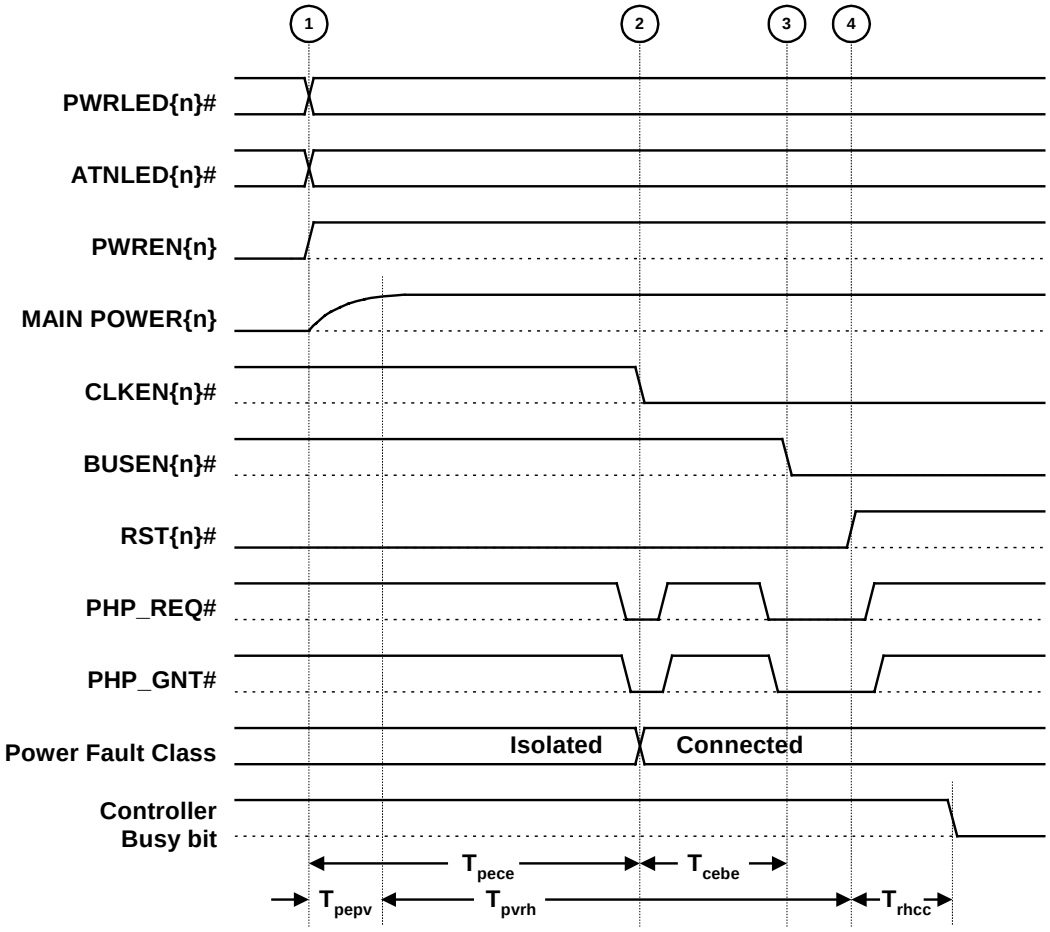



**Figure 3-4: Standard Hot-Plug Controller Slot Enable Command Phases**

**Table 3-5: Standard Hot-Plug Controller Slot Enable Phases**

| Phase | Description | Signals Affected |
|---|---|---|
| 1 | The first phase of a Slot Enable command for slot n causes changes in one or more of the following: **PWREN{n}**, **ATNLED{n}#**, and/or **PWRLED{n}#**. No bus arbitration is required for this phase. If indicator state changes have been requested with a Slot Enable command, the indicator changes occur during this phase. | **PWREN{n}, ATNLED{n}#, PWRLED{n}#** |
| 2 | During the second phase of a Slot Enable command, the **CLKEN{n}#** signal asserts. Secondary PCI bus arbitration is required for this phase. During this phase, the slot-specific **CLK{n}** and **M66EN{n}** pin(s) of the slots are connected to the active bus segment. (See Figure 3-5 and Figure 3-6.) | **CLKEN{n}#** |
| 3 | During the third phase of a Slot Enable command, the **BUSEN{n}#** signal asserts. Secondary PCI bus arbitration is required for this phase. After bus connection, but before the deassertion of **RST{n}#**, **REQ64#** and a PCI-X initialization pattern (if the secondary bus supports PCI-X) must be driven onto the secondary bus. (See Figure 3-7 and Figure 3-8.) | **BUSEN{n}#** |
| 4 | During the fourth phase of a Slot Enable command, the **RST{n}#** signal deasserts. This phase occupies the same bus arbitration cycle as phase 3 so that the setup time requirement for the initialization pattern is guaranteed. | **RST{n}#** |

**Table 3-6: Standard Hot-Plug Controller Slot Enable Timing Specifications**

| Timing Parameter | Description | Min | Max | Units |
|---|---|---|---|---|
| $T_{pece}$ | Delay from **PWREN{n}** asserted to **CLKEN{n}#** asserted. (Note 3) | $25+T_{al}$ (Note 1) | $220+T_{al}$ (Note 1) | ms |
| $T_{cebe}$ | Delay from **CLKEN{n}#** asserted to **BUSEN{n}#** asserted. (Note 4) | $25+T_{al}$ (Note 1) | $220+T_{al}$ (Note 1) | ms |
| $T_{pepv}$ | Delay from **PWREN{n}** asserted to Power Valid. (Note 2) | 1.4 | 220 | ms |
| $T_{pvrh}$ | $T_{pvrh}$ is the delay from power valid to **RST#** high. (Its value is specified in PCI 2.2, Section 4.3.2.) | 100 (ref) | | ms |
| $T_{rhcc}$ | Delay from **RST{n}#** high to Command Completion. | 0 | | ms |
| $T_{alto}$ | Arbitration latency timeout (see Section 3.4.1). | | $2^{23}$ | secondary bus clocks |

Notes:
1. The value of $T_{al}$ (arbitration latency) is unspecified. (See Section 3.4.1.)
2. $T_{pepv}$ is used by the Power Controller to control inrush current to the add-in card. (See PCI HP 1.1, Section 3.3.) These values have been derived from the allowable minimum and maximum slew rates specified in PCI HP 1.1.
3. $T_{pece}$ must always be greater than or equal to $T_{pepv}$.
4. The sum of $T_{pece}$ and $T_{cebe}$ (controlled by the SHPC) less $T_{pepv}$ must ensure the slot is in compliance with $T_{pvrh}$.

### Implementation Note: Timer Prescalers

The ranges of values specified for $T_{pece}$ and $T_{cebe}$ have been chosen to enable the SHPC designer to assume that the input clock is running at 25 MHz and select the timer prescaler values to meet the maximum specification. These same prescaler values yield times that meet the minimum specification if the clock source is running at 200 MHz.

Figure 3-5 through Figure 3-8 illustrate the bus timing requirements for bus arbitration and stabilization during Slot Enable command. It is recommended that the **PAR** signal is driven low and the other bus control signals (**FRAME#, IRDY#**, **TRDY#**, **DEVSEL#**, and **STOP#**), **AD[31::00]**, and **C/BE [3::0]#** are driven high by the SHPC during these phases, except as required for the PCI-X initialization pattern. Table 3-7 defines the recommended timing of bus control signals for phases 2 through 4 of the Slot Enable commands.

### Implementation Note: Reducing Bus Switch Charge Transfer

It is recommended that the SHPC drive the secondary bus signals high during a slot enable sequence that connects the bus. This is done to reduce the amount of charge transfer through the bus switches (when connection occurs) between the secondary bus and the pre-charged (high logic state) slot-side signals. When the bus is in PCI-X mode, a PCI-X initialization pattern must be driven on the **FRAME#**, **IRDY#**, **TRDY#**, **DEVSEL#**, and **STOP#** signals. Normally, the control signals are pulled up. However, driving the control signals high with a low impedance output buffer adds noise margin to the bus. The **AD[31::00]** and **C/BE [3::0]#** signals are also driven high since PCI 2.2, Section 3.4.3, requires the bus to be parked within eight clocks when the bus is idle. The **PAR** signal is driven low to maintain even parity on the bus. If **FRAME#** signal integrity is corrupted and is seen asserted during the bus switch, a parity error will not be detected.

### Implementation Note: Clock Distribution

A PCI clock driver technique often used by Platform designs is to connect the outputs of multiple buffers together in order to minimize the clock skew seen at the PCI slots. This practice is discouraged for SHPC Platforms because glitches occurring on one hot-plug slot are seen by other slots. If the **CLK** signal at one slot violates the PCI 2.2 or PCI-X 1.0a specification, the bus protocol state machines on enabled cards are potentially disrupted.

An SHPC that supports 64-bit bus segments is required to supply a **REQ64#** signal. For 64-bit slots, the **REQ64{n}#** signal is connected to the bridge **REQ64#** output when **BUSEN{n}#** is asserted. For 32-bit slots that are implemented on 64-bit bus segments, the Platform designer is responsible for the termination of the **REQ64{n}#** signal, and the **REQ64{n}#** signal is not connected to any other slots or bridge, as described in PCI 2.2, Section 3.8.1. On 64-bit slots, **AD[63::32]**, **C/BE[7::4]#**, and **PAR64** are pulled up or “kept” as described in PCI 2.2, Section 3.8.1.

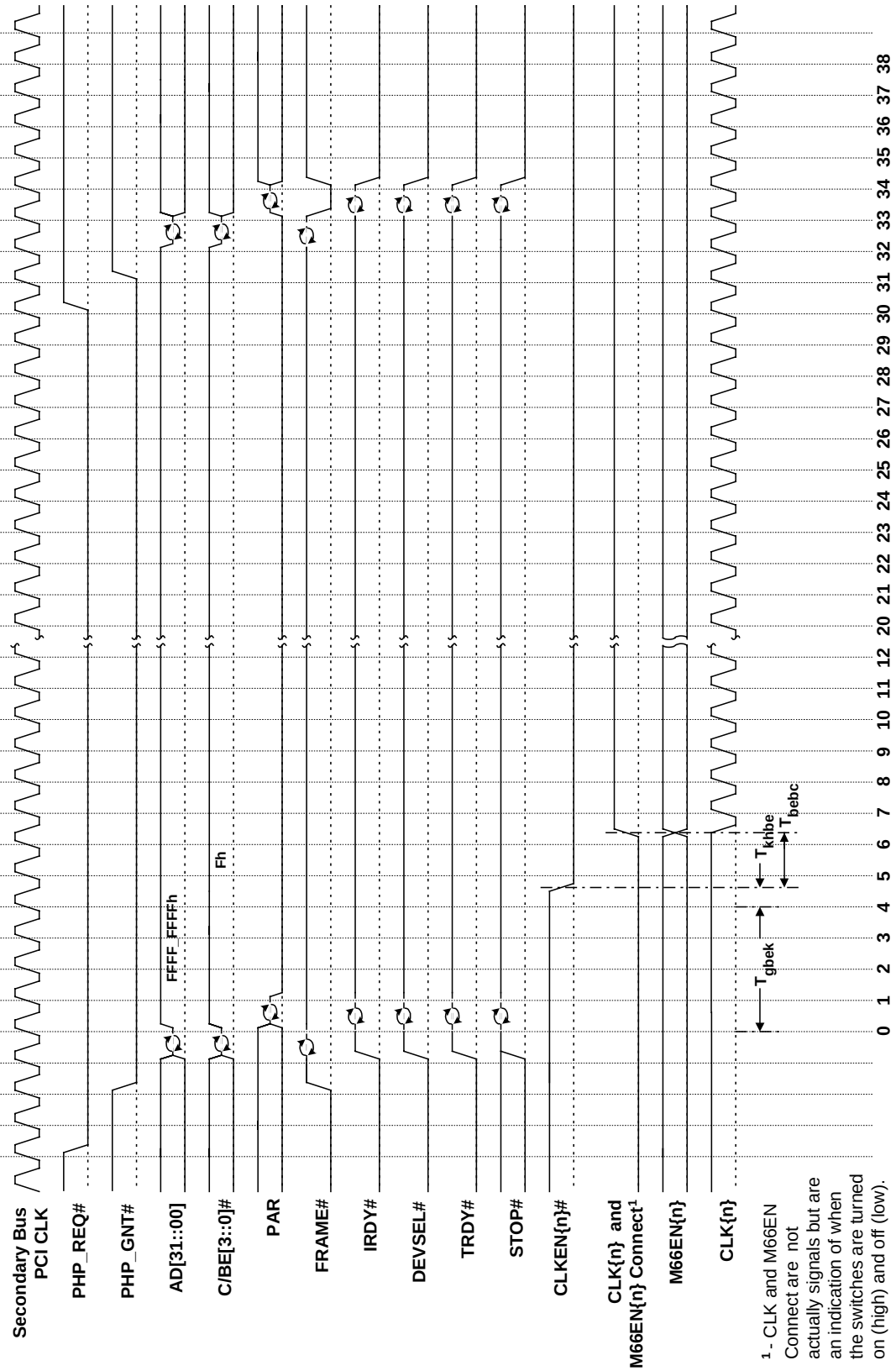



**Figure 3-5: Conventional Mode SHPC Slot Enable Sequence Bus Stabilization Timing, Phase 2**

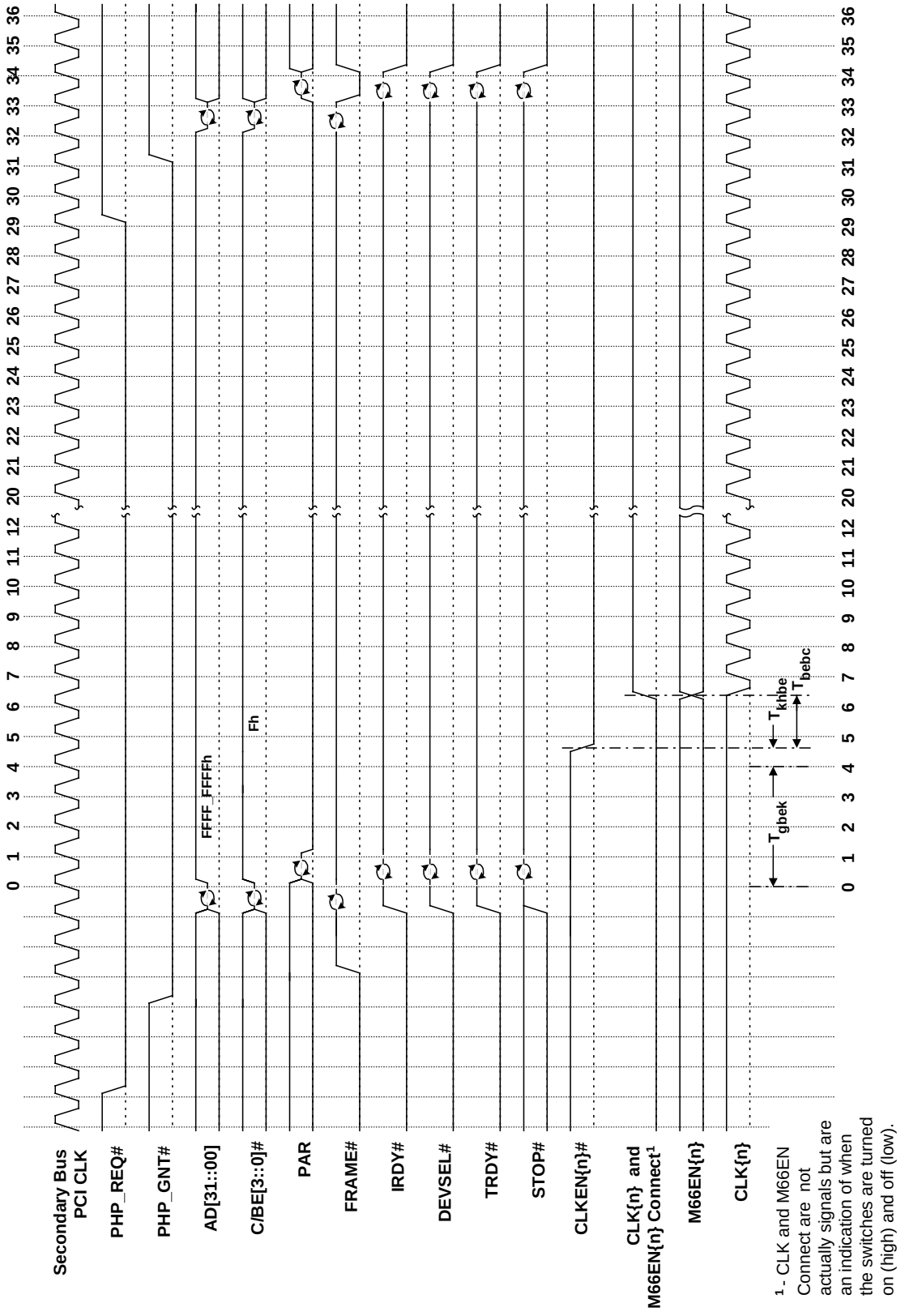



**Figure 3-6: PCI-X Mode SHPC Slot Enable Sequence Bus Stabilization Timing, Phase 2**

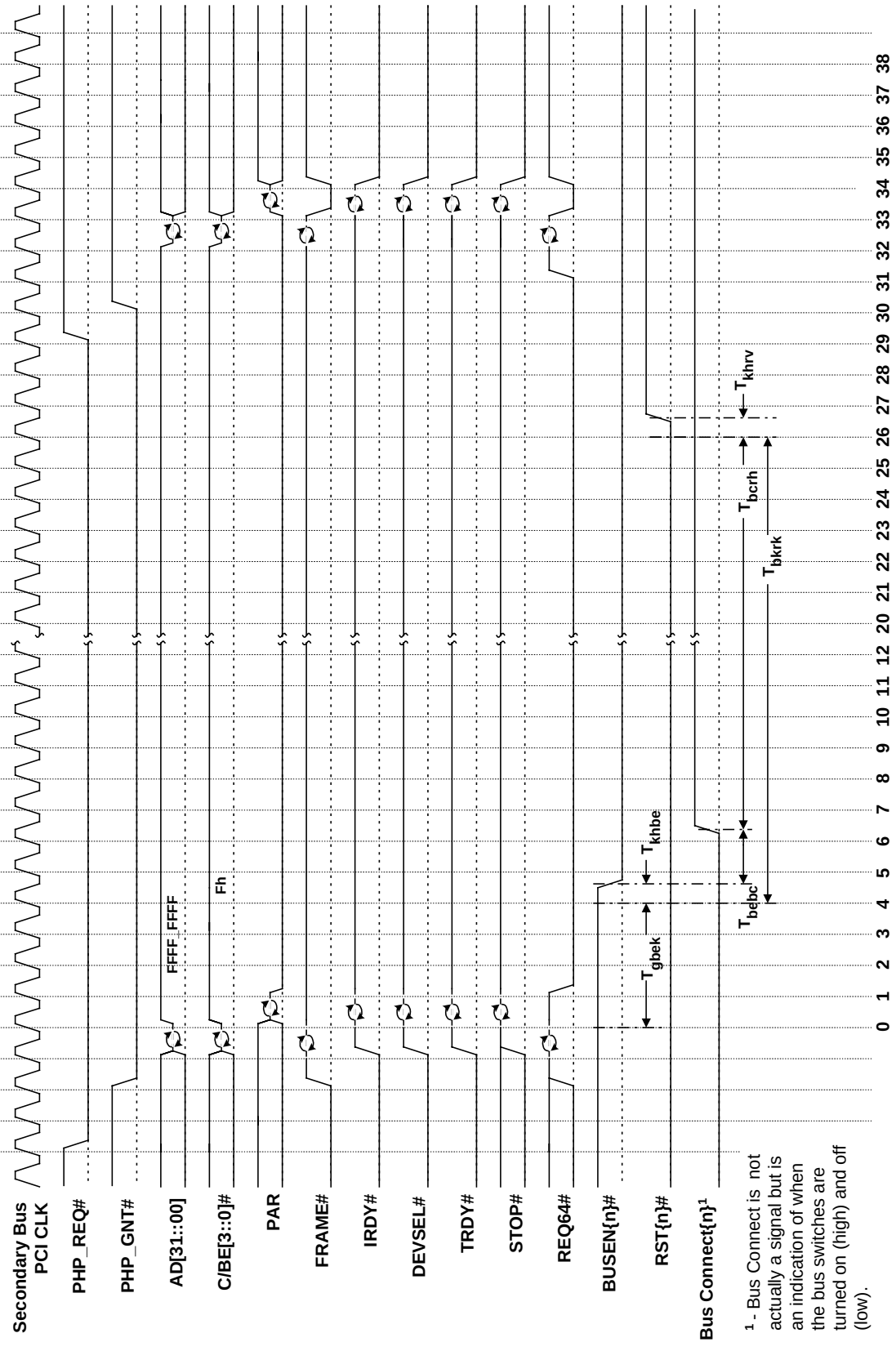



**Figure 3-7: Conventional Mode SHPC Slot Enable Sequence Bus Stabilization Timing, Phases 3 and 4**

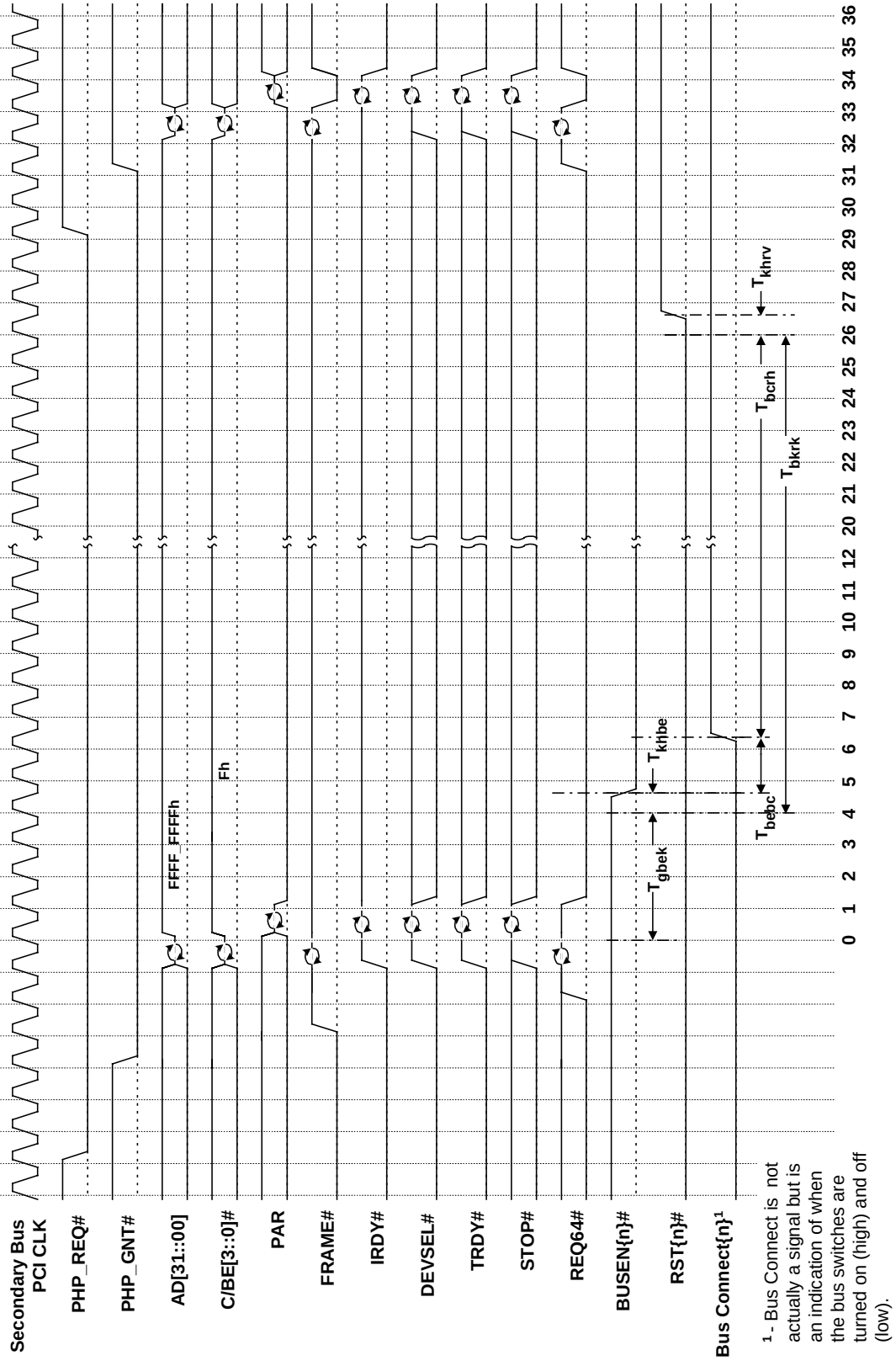



**Figure 3-8: PCI-X Mode SHPC Slot Enable Sequence Bus Stabilization Timing, Phases 3 and 4**

**Table 3-7: SHPC Slot Enable and Disable Command Execution Sequence Bus Stabilization Timing**

| Timing Parameter | Description | Min | Max | Units | Reference Figures | Affected Components |
|---|---|---|---|---|---|---|
| $T_{gbek}$ | Delay from bus idle-grant to **CLKEN{n}#** or **BUSEN{n}#** change | 4 | 4 | Secondary Bus Clocks | Figure 3-5<br>Figure 3-6<br>Figure 3-7<br>Figure 3-8<br>Figure 3-12<br>Figure 3-13 | SHPC |
| $T_{khbe}$ | Delay from secondary CLK to **CLKEN{n}#** or **BUSEN{n}#** change – output propagation delay | 0 | 20 | ns | Figure 3-5<br>Figure 3-6<br>Figure 3-7<br>Figure 3-8<br>Figure 3-10<br>Figure 3-11<br>Figure 3-12<br>Figure 3-13 | Slot Control Logic |
| $T_{bebc}$ | Delay from **BUSEN{n}#** or **CLKEN{n}#** changed to **M66EN{n}**, bus signal for slot n or **CLK{n}** connection or disconnection | 7 | 60 | ns | Figure 3-5<br>Figure 3-6<br>Figure 3-7<br>Figure 3-8<br>Figure 3-10<br>Figure 3-11<br>Figure 3-12<br>Figure 3-13 | Bus Switches |
| $T_{bcrh}$ | Delay from bus connection to **RST{n}#** deassertion (Note) | 10 | 22 | Secondary Bus Clocks | Figure 3-7<br>Figure 3-8 | SHPC |
| $T_{khrv}$ | Delay from **CLK** to **RST{n}#** change - output propagation delay | 0 | 20 | ns | Figure 3-7<br>Figure 3-8<br>Figure 3-10<br>Figure 3-11<br>Figure 3-15<br>Figure 3-16 | Slot Control Logic |
| $T_{bkrk}$ | Delay from **BUSEN{n}#** asserted to **RST{n}#** deasserted and delay from **RST{n}#** asserted to **BUSEN{n}#** deasserted | 22 | 22 | Secondary Bus Clocks | Figure 3-7<br>Figure 3-8<br>Figure 3-10<br>Figure 3-11 | SHPC |
| $T_{grlk}$ | Delay from bus idle-grant to **RST{n}#** asserted | 4 | 4 | Secondary Bus Clocks | Figure 3-10<br>Figure 3-11 | SHPC |

Note: PCI 2.2 and PCI-X 1.0a require some bus signals to be driven 10 clocks before **RST#** deassertion. The SHPC must insure **BUSEN{n}#** is asserted ahead of **RST{n}#** deassertion by 10 clocks plus additional clocks to adjust for bus connection latencies associated with the Slot Control Logic and bus switches ($T_{bebc}$).

### 3.4.4. Slot Disable Sequencing

The SHPC disables a slot in the following cases:

- The SHPC receives a Slot Disable command from the software (see Section 4.6.1).
- The slot's MRL opens while the slot is enabled (see Section 4.8).

To disable the slot, the SHPC switches **PWREN{n}**, **CLKEN{n}#**, **BUSEN{n}#**, and **RST{n}#** signals as described in this section. The SHPC also switches **PWRLED{n}#** and **ATNLED{n}#** signals as described in this section, if the command from the software requires it.

The SHPC executes the Slot Disable command in four phases. Slot control signals are switched during each phase, as shown in Figure 3-9 and defined in Table 3-8. Table 3-9 defines the recommended timing between these phases. See Section 4.8 for a description of control signals affected when the opening of an MRL disables a slot.

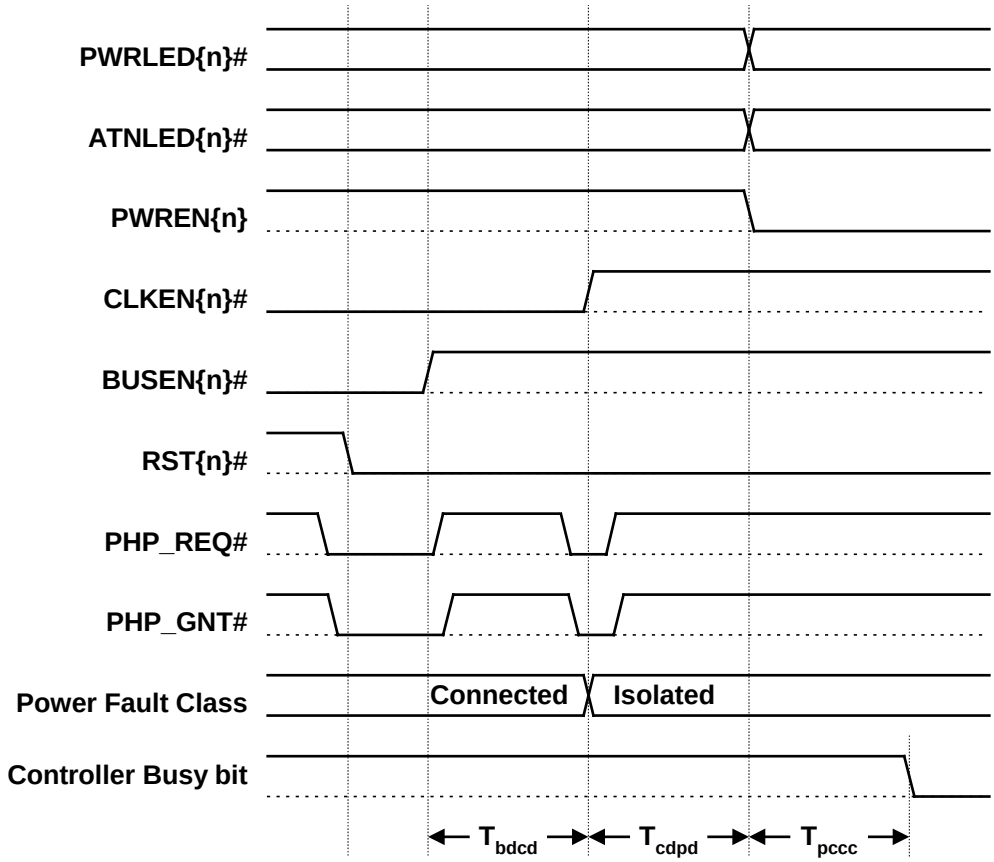



**Figure 3-9: Standard Hot-Plug Controller Slot Disable Phases**

**Table 3-8: Standard Hot-Plug Controller Slot Disable Phases**

| Phase | Description | Signals Affected |
|---|---|---|
| 1 | During the first phase of a Slot Disable command, the **RST{n}#** signal asserts. Secondary bus arbitration is required for this phase. (See Figure 3-10 and Figure 3-11 for bus stabilization requirements.) | **RST{n}#** |
| 2 | During the second phase of a Slot Disable command, the **BUSEN{n}#** signal is deasserted. Secondary bus arbitration is required for this phase. (See Figure 3-10 and Figure 3-11 for bus stabilization requirements.) | **BUSEN{n}#** |
| 3 | During the third phase of a Slot Disable command, the **CLKEN{n}#** signal deasserts. Secondary bus arbitration is required for this phase. (See Figure 3-12 and Figure 3-13 for bus stabilization requirements.) | **CLKEN{n}#** |
| 4 | During the fourth phase of a Slot Disable command, **PWREN{n}** deasserts. No bus arbitration is required for this phase. If indicator state changes have been requested with a Slot Disable command, the indicator state changes occur during this phase. (See Section 3.4.5, and Figure 3-14 for additional timing requirements.) | **PWREN{n}, ATNLED{n}#, PWRLED{n}#** |

**Table 3-9: Standard Hot-Plug Controller Slot Disable Timing Specifications**

| Timing Parameter | Description | Min | Max | Units |
|---|---|---|---|---|
| $T_{bdcd}$ | Delay from **BUSEN{n}#** deasserted to **CLKEN{n}#** deasserted. | 4 | $100+T_{al}$ (Note) | µs |
| $T_{cdpd}$ | Delay from **CLKEN{n}#** deasserted to **PWREN{n}**, deasserted. | 4 | 100 | µs |

Note: The value of $T_{al}$ (arbitration latency) is unspecified. (See Section 3.4.1.)

Figure 3-10 through Figure 3-13 illustrate the bus timing requirements for bus arbitration and stabilization during Slot Disable commands. Table 3-7 defines the recommended timing of bus control signals for phases 1, 2, and 3. To simplify the implementation, the four phases for disabling a slot shown in Figure 3-9 are the inverse of the four phases for enabling a slot shown in Figure 3-4.

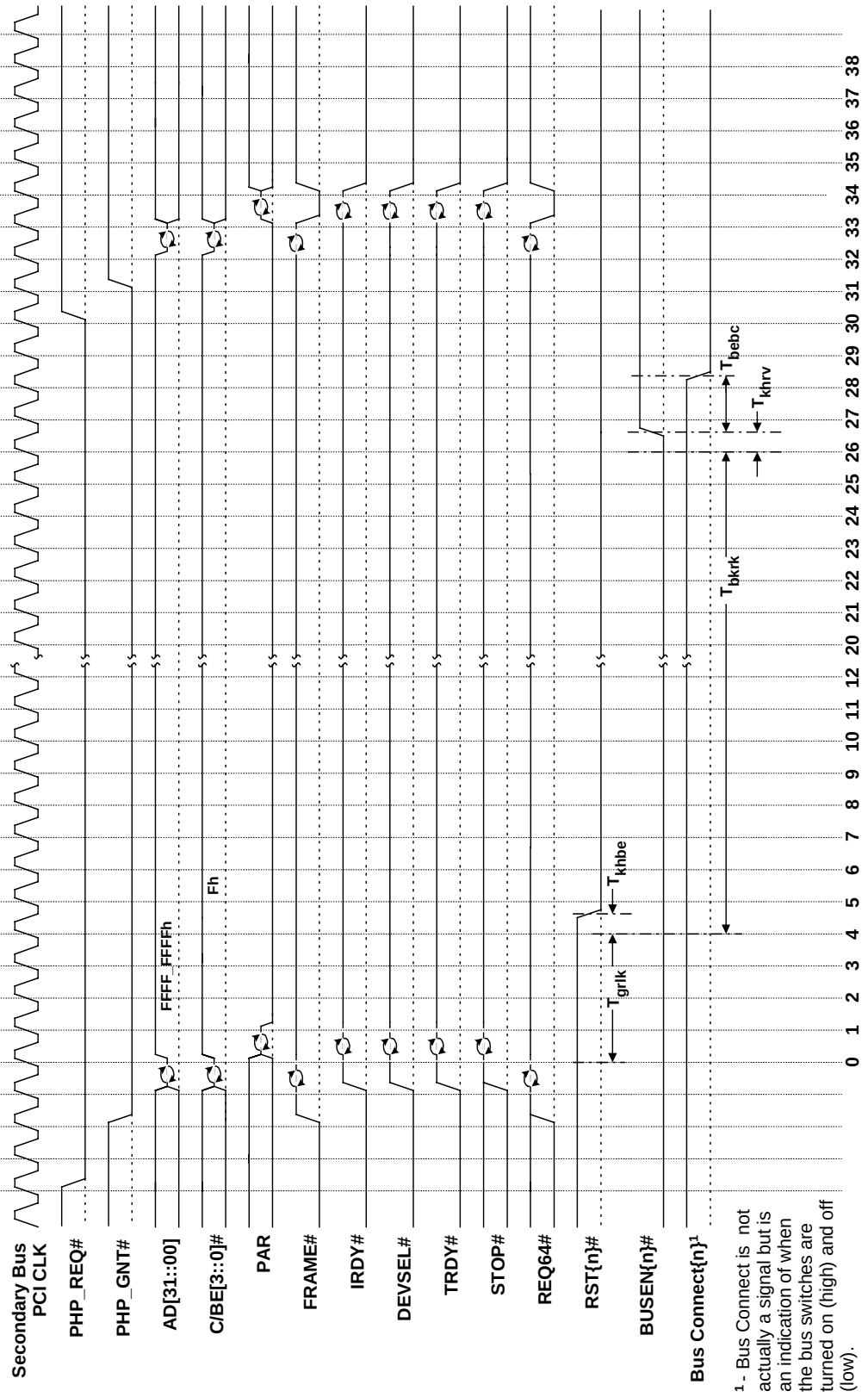



**Figure 3-10: Conventional Mode SHPC Slot Disable Sequence Bus Stabilization Timing, Phases 1 and 2**

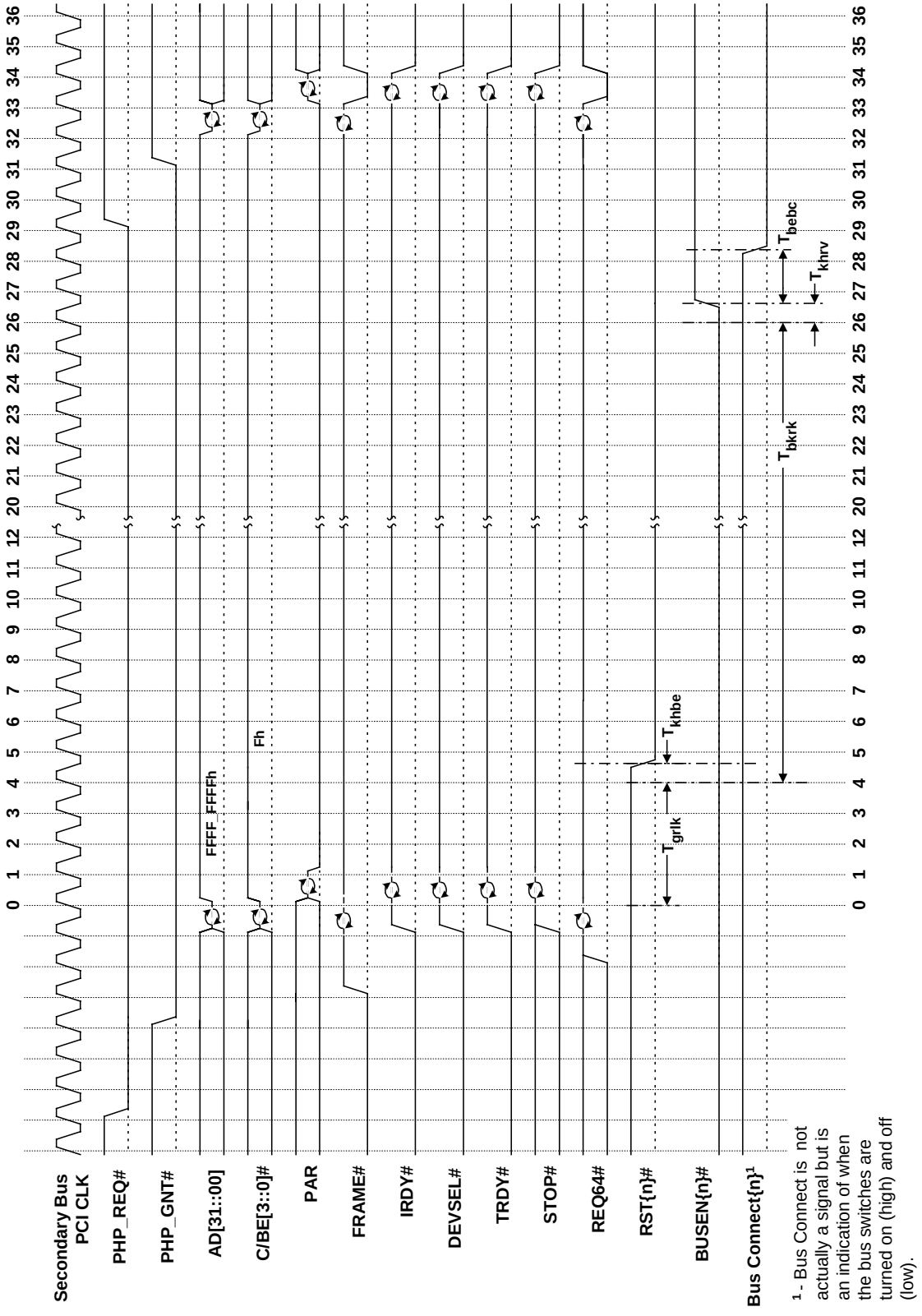



**Figure 3-11: PCI-X Mode SHPC Slot Disable Sequence Bus Stabilization Timing, Phases 1 and 2**

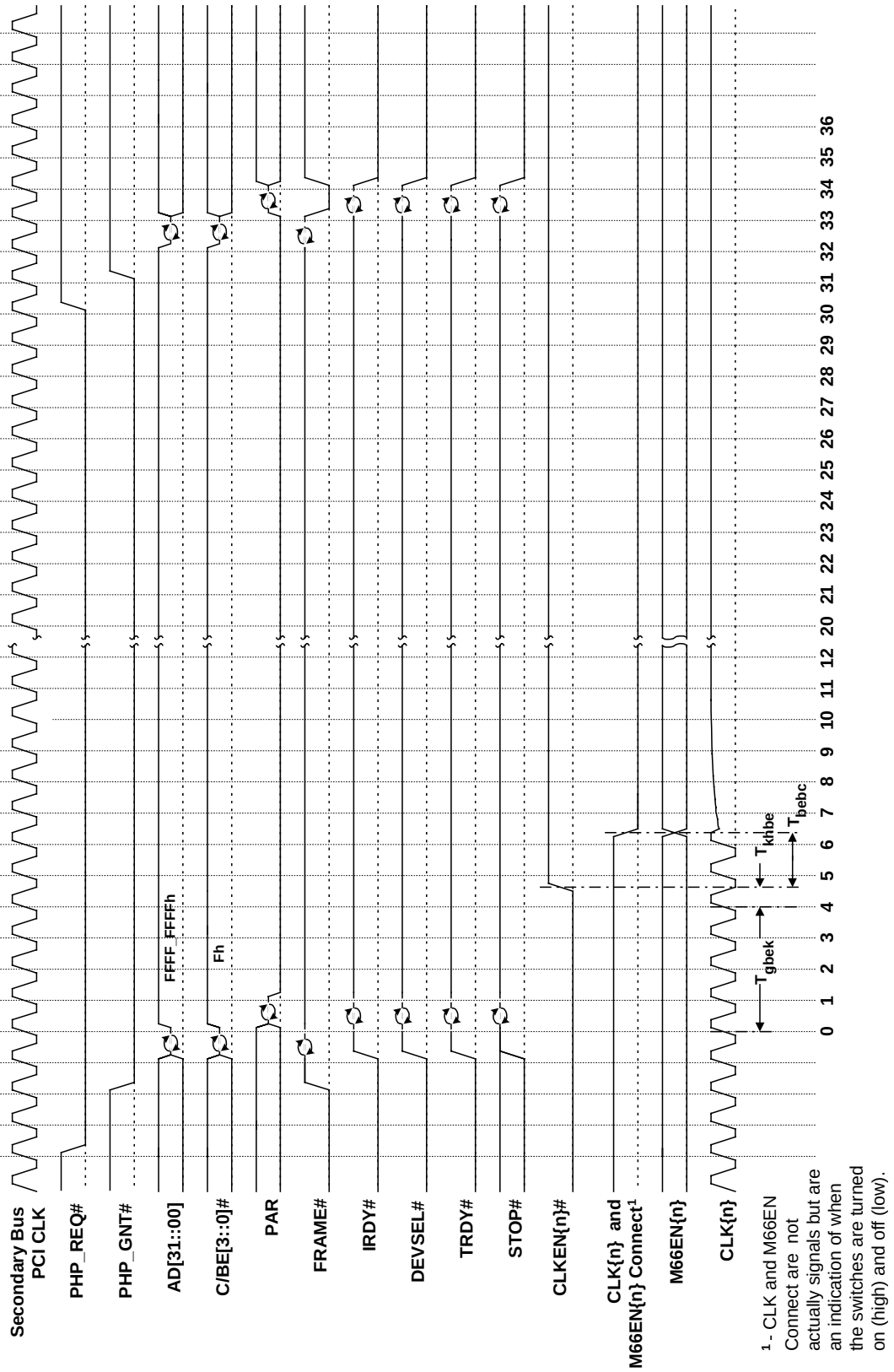



**Figure 3-12: Conventional Mode SHPC Slot Disable Sequence Bus Stabilization Timing, Phase 3**

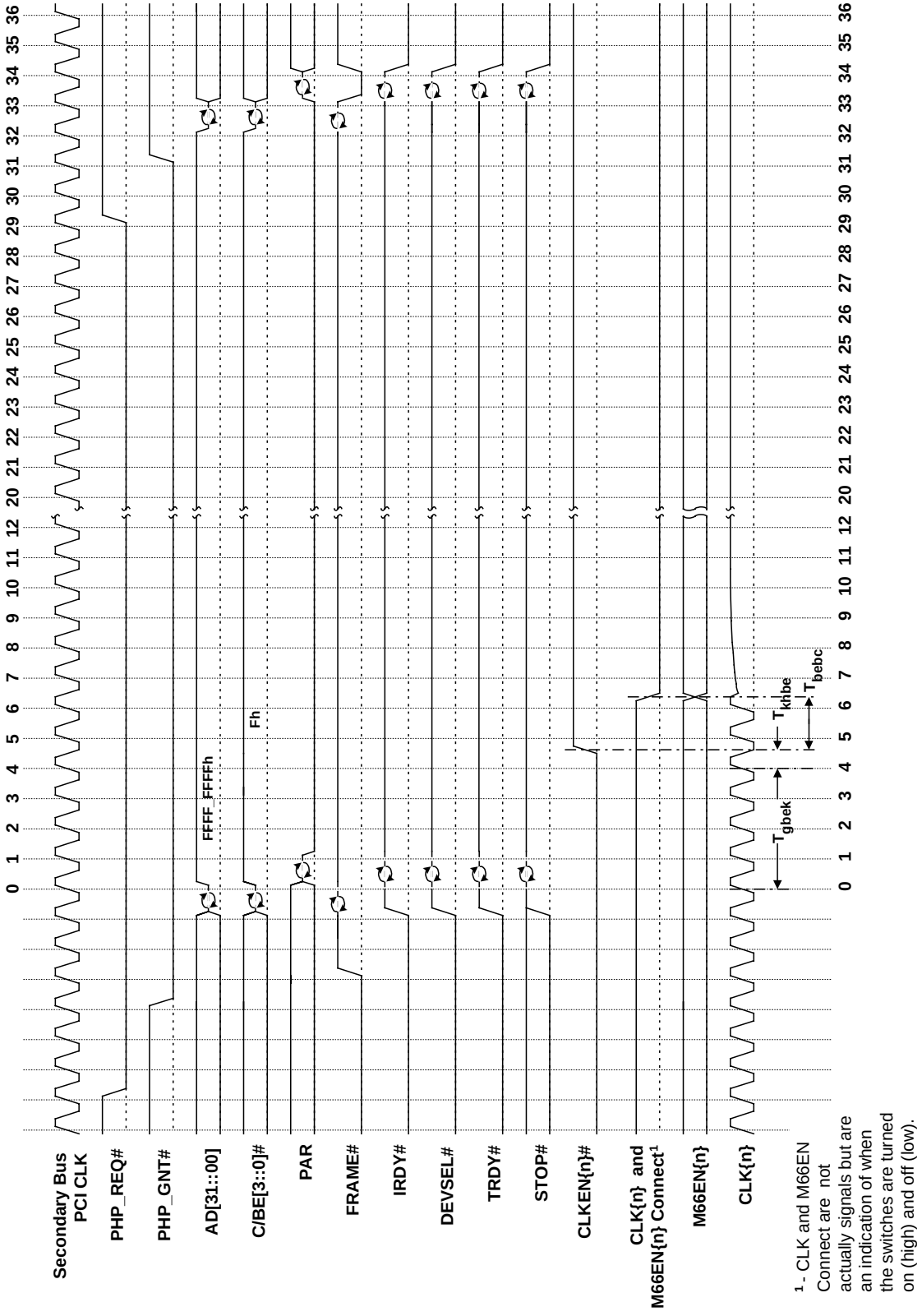



**Figure 3-13: PCI-X Mode SHPC Slot Disable Sequence Bus Stabilization Timing, Phase 3**

### 3.4.5. Single-phase Operations

The single-phase commands are:

- Slot Power Only
- Power-Only All Slots (if the SHPC is designed to power all slots simultaneously)
- Power Indicator ON/OFF/BLINK (if no slot state change is requested)
- Attention Indicator ON/OFF/BLINK (if no slot state change is requested)
- Set Bus Segment Speed/Mode (used to deassert all **RST{n}#**)
- Vendor-Specific Commands (as defined by the SHPC vendor)

If a command is issued to simultaneously change the state of an indicator and Slot Power Only, both operations are performed (simultaneously) using a single-phase controller sequence.

The SHPC also autonomously performs a single-phase operation to change the state of **RST{n}#** following a bus reset event (see Table 3-10), and to toggle blinking indicators. The Controller Busy bit must remain cleared during such autonomous operations (unless software writes to the Controller Command register), even if the SHPC uses the command state machines to perform these operations.

Single-phase SHPC commands affect one or more of the following slot-specific control signals:

- **PWREN{n}**
- **ATNLED{n}#**
- **PWRLED{n}#,**
- **RST{n}#**

Table 3-10 summarizes the slot-state transitions that are possible using a single-phase operation.

**Table 3-10: Single-phase State Transitions**

| Current Slot State | New Slot State | Stimulus |
|---|---|---|
| Off | Powered | • Power slot command |
| Powered | Off | • Slot disable command<br>• MRL opened |
| Indicator On/Off | Indicator Off/On | • Change state of indicator output (toggle) required to maintain blink frequency<br>• Indicator On/Off/Blink command |
| **RST{n}#** Asserted (Note) | **RST{n}#** Deasserted | • Clearing Secondary Bus Reset in the Bridge Control register<br>• Completion of Set Frequency/Mode command<br>• Primary **RST#** deasserts |

Note: See Section 3.1.4 for the assertion of **RST{n}#**.

**Table 3-11: Single-phase Timing Specifications**

| Timing Parameter | Description | Min | Max | Units |
|---|---|---|---|---|
| $T_{pccc}$ | Delay from **PWREN** change to command completion. (See Figure 3-9 and Figure 3-14.) | 25 (Note 1) | 220 (Note 2) | ms |
| $T_{grhk}$ | Delay from bus idle-grant to **RST{n}#** deassertion. (See Figure 3-15 and Figure 3-16.) | 26 | 26 | Secondary Bus Clocks |

Notes:

1. $T_{pccc}$ min is required to be greater than the supply voltage rail rise (for power up) and fall (for power down) time for worst-case add-in card decoupling capacitances (see PCI HP 1.1).
2. This value corresponds to the maximum voltage rail rise time in the power up case. Larger values of $T_{pccc}$ are allowed in the power down case, within the limits of the maximum command completion time specified in Section 4.6.

When executing a Slot Power Only command, the SHPC must provide a power rail voltage settling time before completing the command as illustrated in Figure 3-14. Recommended timing is defined in Table 3-11. In the case of a Slot Power Only command, the timing parameter $T_{pccc}$ is required to be large enough to insure that all main supply rails have fully powered up to within specifications defined by PCI 2.2, Section 4.3.4.1.

When a slot or slots are in the powered-only or enabled state and a Slot Disable command is executed, the SHPC must provide the equivalent supply voltage fall time before completing the command. In the case of a Slot Disable command, the timing parameter $T_{pccc}$ is required to be large enough to insure that all main supply rails have fully powered down to within $V_{off}$ volts.

### Implementation Note: $T_{pccc}$ Specification for Slot Disable Commands

When the slot is turned off, the voltage at the slot does not instantaneously diminish to zero. If the slot pins have residual voltage when the add-in card is removed, there is a potential for arcing on the connector. In addition, when back-to-back Slot Disable and Slot Enable commands are executed, there is a potential that the supply rail voltage will not diminish completely to zero before software enables the slot. If this occurs, the card may not initialize properly.

When a single-phase command is executed that changes both the state of an indicator and the state of slot power, the output controls to the slot are permitted to change simultaneously as shown in Figure 3-14.

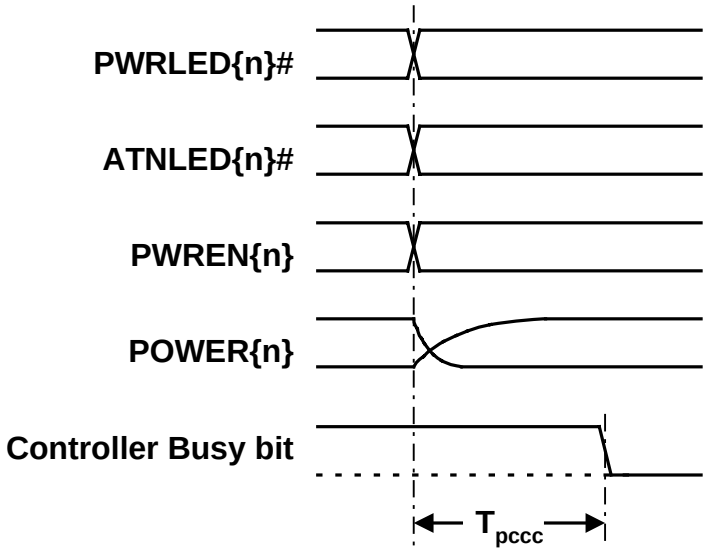



**Figure 3-14: Slot Power Only Command Execution**

Arbitration is required when deasserting **RST{n}#** so that an appropriate initialization pattern can be driven as illustrated in Figure 3-15 and Figure 3-16. Recommended timing is defined in Table 3-7 and Table 3-11.

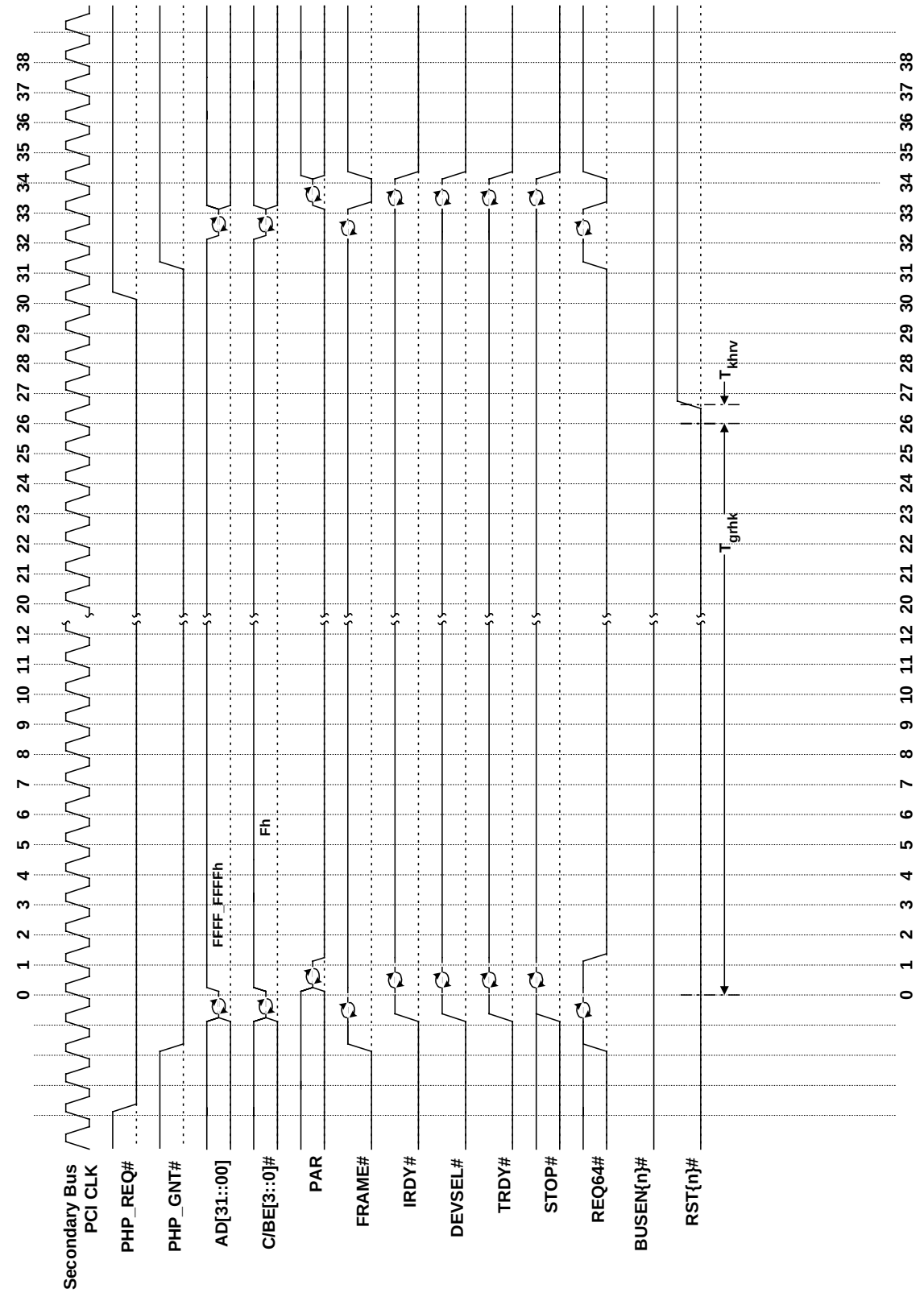



**Figure 3-15: Conventional Mode Deassertion of RST{n}# After a Bus Segment Reset or SHPC Set Frequency/Mode Command**

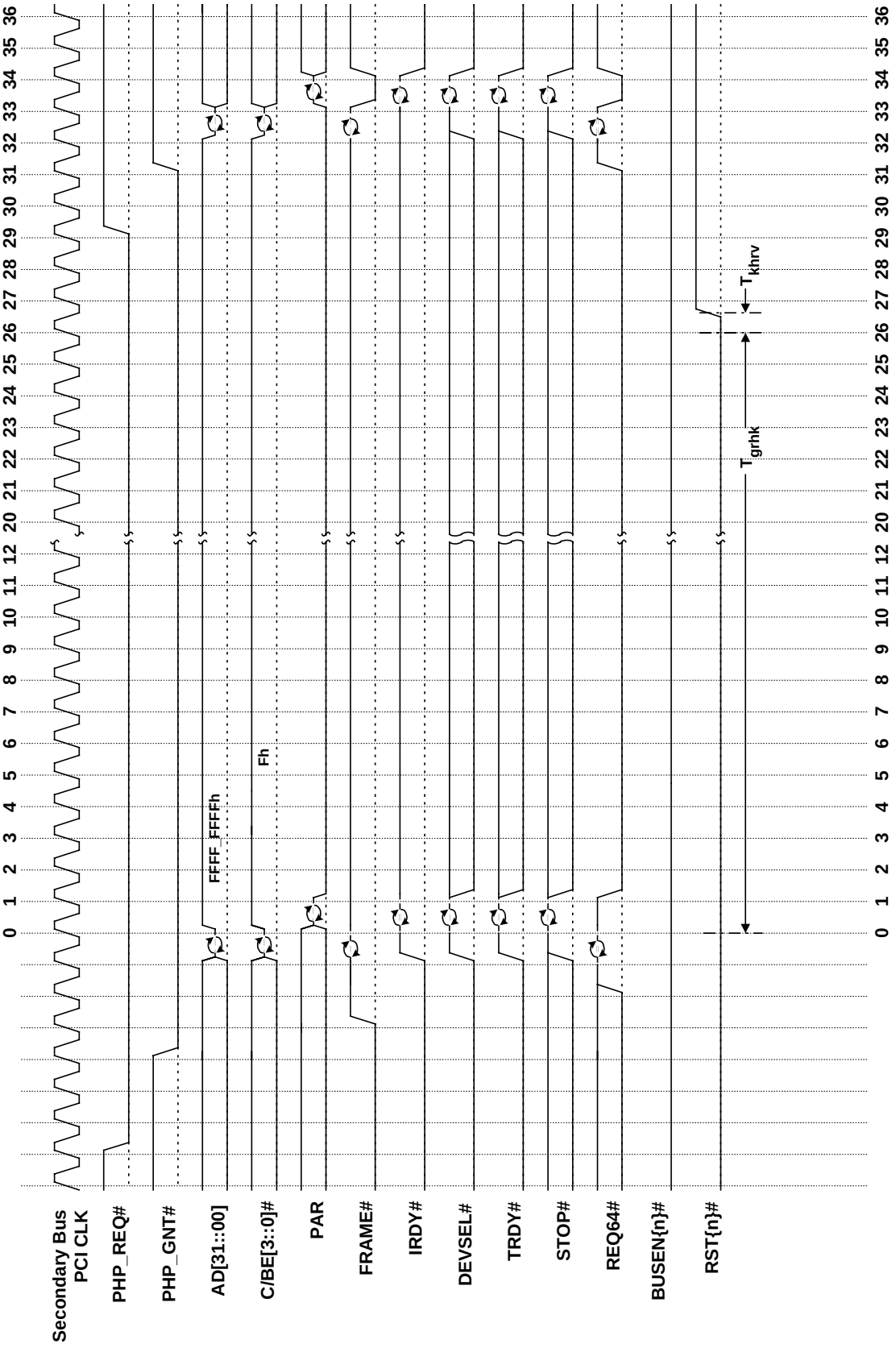



**Figure 3-16: PCI-X Mode Deassertion of RST{n}# After a Bus Segment Reset or SHPC Set Frequency/Mode Command**

### 3.4.6. SHPC and Slot Control Logic Electrical Characteristics

Recommended electrical parameters are listed in Table 3-12 and Table 3-13. These recommendations apply to the interfaces between the slot-specific circuit blocks shown in Figure 3-1 and Figure 3-2. The serial interface between the SHPC and the Slot Control Logic is beyond the scope of this specification. Table 3-14 and Table 3-15 list recommended DC and AC specifications for the bus switches. See PCI-X 1.0a or PCI 2.2 for the electrical requirements of the following signals:

- **RST#**
- **CLK**
- **PRSNT1#**
- **PRSNT2#**
- **PCIXCAP**
- **M66EN**

**Table 3-12: Recommended SHPC and Slot Control Logic DC Electrical Characteristics**

| Parameter | Test Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| **Slot Control Logic signals: PWREN{n}, BUSEN{n}#, CLKEN{n}#, ATTLED{n}#, and PWRLED{n}#.** | | | | | |
| $V_{ih}$ – High-level input voltage (Note 2) | | 2.0 | | | V |
| $V_{il}$ – Low-level input voltage (Note 2) | | | | 0.8 | V |
| $V_{oh}$ – High-level output voltage | | 2.4 | 3.4 | | V |
| $V_{ol}$ – Low-level output voltage | | | 0.2 | 0.4 | V |
| Slot Control Logic input pin leakage | $V_i$ = 0 through 5.5 V, internal pull-up/pull-down resistors disabled (if used) | | | ±70 | µA |
| Input capacitance | $V_i$ = 0 or 5.5 V | | | 10 | pF |
| **Bus Switch to Slot Control Logic I/F** | | | | | |
| **BUSEN{n}#** $I_{oh}$ high-level output current (Note 1) | | -8 | | | mA |
| **BUSEN{n}#** $I_{ol}$ low-level output current (Note 1) | @ $V_{ol}$ = 0.4 V | 16 | | | mA |
| **CLKEN{n}#** $I_{oh}$ high-level output current | | -4 | | | mA |
| **CLKEN{n}#** $I_{ol}$ low-level output current | | 4 | | | mA |
| **Slot Control Logic to Power Controller I/F** | | | | | |
| **PWREN{n}** output $I_{oh}$ high-level output current | | -4 | | | mA |
| **PWREN{n}** output $I_{ol}$ low-level output current | | 4 | | | mA |
| **User Controls (inputs) and Indicators (outputs)** | | | | | |
| **MRLSW{n}#**, **BUTTON{n}#** hysteresis, $\Delta V_T$ (See Appendix A.) | | 0.15 | | 2.0 | V |
| **PWRLED{n}#**, **ATTLED{n}#** $I_{ol}$ Low-Level Output Current (Note 3) | @ $V_{ol}$ = 0.4 V | 24 | | | mA |
| **Power Controller Specifications** | | | | | |
| $V_{off}$ – Main supply rail voltage $T_{pccc}$ after **PWREN{n}** deassertion | Power Controller **PWREN{n}** input $V_{il}$ = 0.8 V | -100 | | +100 | mV |

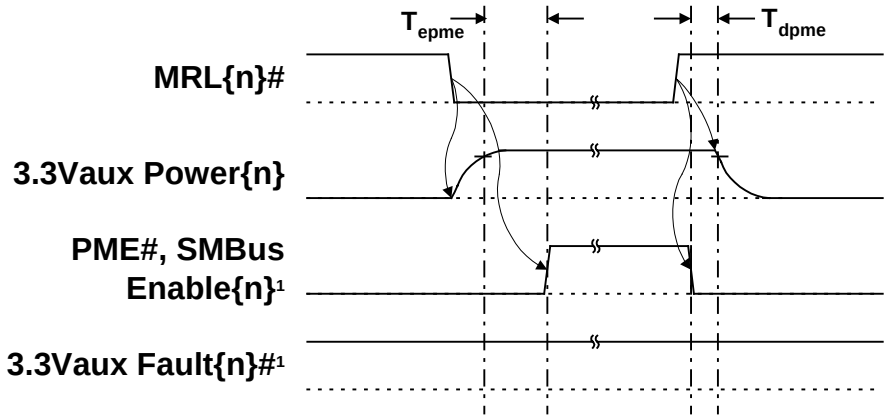
Notes:
1. **BUSEN{n}#** must drive the enable pin on many bus switch devices. Hence, a higher drive current is recommended to satisfy timing requirements.

2. It is permitted for the slot-side interface outputs (**PWREN{n}**, **BUSEN{n}#**, **CLKEN{n}#**, **ATTLED{n}#**, and **PWRLED{n}#**) to support a diagnostic read-back and/or manufacturing test capabilities and, in such cases, input specifications for these pins are required.
3. The output drive current is strong enough to directly drive two LED indicators.

**Table 3-13: Recommended SHPC and Slot Control Logic AC Switching Characteristics**

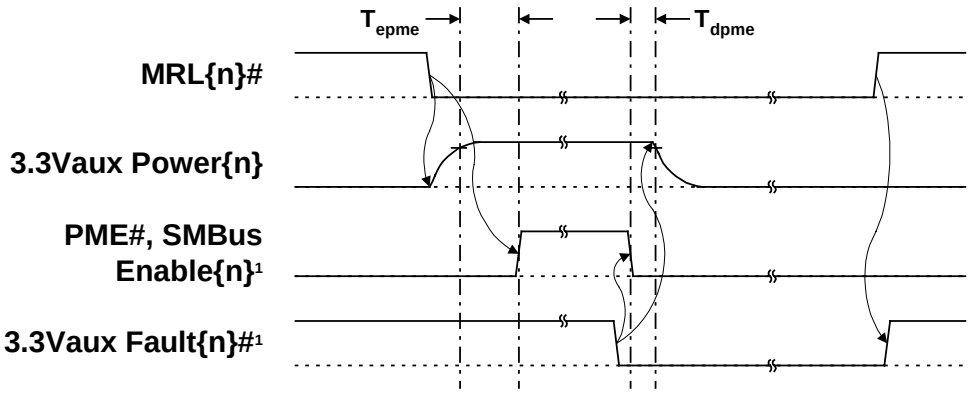
| Parameter | Test Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| $T_{rf}$ – input rise and fall times (Note 2) | 10% to 90% of $V_{CC}$ (Note 1) | 3 | | 6 | ns |
| $T_{mSL}$ – minimum switch latency (bounce immunity period) for **MRL{n}#** and **BUTTON{n}#**, **PRSNT1{n}#**, **PRSNT2{n}#**, **PCIXCAP{n}**, and **M66EN{n}** inputs. (See Appendix A.) | | 8.0 | | | ms |
| $T_{epme}$ – Time delay from **3.3Vaux{n}** rail in-specification to **PME{n}#** and SMBus connection | | 10.0 | | | ms |
| $T_{dpme}$ – Time delay from **PME{n}#** and SMBus disconnection to **3.3Vaux{n}** rail out of specification | | 1.0 | | | ms |

Notes:
1. This parameter applies to the I/O buffer supply rail regardless of the operating voltage.
2. Applies to all inputs except **MRL{n}#** and **BUTTON{n}#**, **PRSNT1{n}#**, **PRSNT2{n}#**, **PCIXCAP{n}**, and **M66EN{n}**. The exception inputs are expected to have much slower rise and fall times and require hysteresis inputs or debounce circuitry (see Appendix A).




**Note:**
1. See Sections 3.1.2.2 and 3.1.2.3.

**Figure 3-17: PME# and SMBus Signal Connect and Disconnect Timing**




**Note:**
1. See Sections 3.1.2.2 and 3.1.2.3.

**Figure 3-18: 3.3Vaux Fault Behavior**

The parameters in Table 3-14 and Table 3-15 were derived from a survey of industry-standard device specifications and were used in the development of the timing recommendations in this specification.

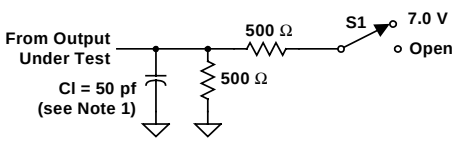
**Table 3-14: Typical Bus Switch DC Specifications**

| Parameter | Test Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| $R_{on}$ | $V_{CC}$ = min., $V_I$ = 0.0 V, $I_{on}$ = 48 mA | | 5 | | Ω |
| | $V_{CC}$ = min., $V_I$ = 2.4 V, $I_{on}$ = 15 mA | | 10 | | Ω |
| $I_{off}$ – leakage current between side A (bus-side) and side B (add-in card slot) when **BUSEN{n}#** is deasserted | | -100 | | 100 | nA |
| $I_{ozb}$ – Side B, high impedance output current (slot-side biasing when **BUSEN{n}#** is deasserted) | $V_{bias}$ = $V_{CC}$ max, $V_{olb}$ = 0.0 V | | | 2 | mA |
| $I_{oza}$ – Side A, high impedance leakage current (bus-side leakage when **BUSEN{n}#** is deasserted) | | | | ±1 | µA |

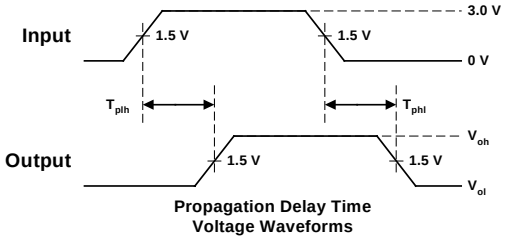
**Table 3-15: Typical Bus Switch AC Switching Characteristics**

| PARAMETER | TEST CONDITIONS | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| $T_{plh}$, $T_{phl}$ | See Figure 3-19. | 10 | | 35 | ns |
| $T_{pzl}$, $T_{plz}$, $T_{pzh}$, $T_{phz}$, | See Figure 3-19. | 1 | | 6 | ns |
| Propagation delay through switch (Note) | | | 0.25 | | ns |

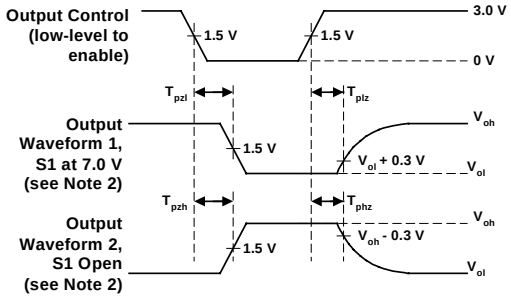
Note: After the bus switch has been enabled, the switch contributes no propagation delay other than the RC delay of the ON resistance of the switch and the load capacitance. The time constant for the switch alone is of the order of 0.25 ns for 50 pF load. This time constant is much smaller than the rise/fall times of typical driving signals, so it adds negligible propagation delay to the system. The propagation delay of the bus switch when used in a system is determined by the signal source circuit on the driving side of the switch and its interaction with the load on the other side of the switch.




| TEST | S1 Position |
|---|---|
| $T_{plh}/T_{phl}$ | Open |
| $T_{pzl}/T_{plz}$ | 7.0 V |
| $T_{pzh}/T_{phz}$ | Open |




Propagation Delay Time
Voltage Waveforms




Enable and Disable Delay
Time Voltage Waveforms

**Notes:**

1. $C_l$ includes probe and jig capacitance.
2. Output Waveform 1 is for an output with internal conditions such that the output is low except when disabled by the output control.
   Output Waveform 2 is for an output with internal conditions such that the output is high except when disabled by the output control.
3. All input pulses are supplied by generators having the following characteristics: $P_{rer\ rate} \leq 10$ MHz, $Z_o = 50\ \Omega$, $T_{rise} \leq 25$ ns, and $T_{fall} \leq 25$ ns.
4. The outputs are measured one at a time with one transition per measurement.

**Figure 3-19: Bus Switch Test Load Circuit**

# 3.5. SHPC Hardware Requirements

This section describes SHPC features that are required.

## 3.5.1. SHPC Initialization Requirements

This section describes the various stimuli that initialize portions of a bridge integrated with an SHPC and specifies which portions are affected by which stimuli. When registers are initialized, they return to their default values specified in Sections 4.3, 4.5, PCI 2.2, PCI Bridge 1.1, and PCI-X 1.0a. When state machines are initialized they return to a benign and stable state. In the case of a bridge that is integrated with an SHPC but is installed in an application in which the SHPC is disabled because it does not control any hot-plug slots (as described in Sections 4.3.1 and 4.3.2.4), the bridge is exempt from these requirements. Such a bridge is initialized exclusively as described in PCI 2.2, PCI Bridge 1.1, and PCI-X 1.0a.

### *3.5.1.1. Initialization Stimuli*

There are three stimuli that reset various portions of a bridge and its SHPC.

1. **Primary reset.** PCI 2.2, PCI Bridge 1.1, and PCI-X 1.0a require that **RST#** on the primary bus reset a PCI-to-PCI bridge's primary interface and Configuration Space registers, and that the signal propagate to the secondary bus **RST#**.

   For a host bridge, this is a host-bus reset, or its equivalent, that initializes the host bridge configuration registers and causes the assertion of secondary **RST#**.

   Primary reset asserts any time the supply voltages in the system are not within the specified tolerances. In some systems, primary reset also asserts for other conditions that require restarting the entire system (for example, recovery from a system crash), when some or all of the system is awakened from a low-power state, or under other conditions controlled by the system software.

2. ***D3*-to-*D0* transition.** Bridge and SHPC state information and logic required to enable and assert the Wakeup Signal is defined to be the Wakeup Context for that bridge/SHPC. See Section 6.2.3 for a full definition of Wakeup Context.

3. **Secondary reset.** Secondary reset is internally generated by the bridge. PCI Bridge 1.1 requires a PCI-to-PCI bridge to assert secondary **RST#** whenever primary **RST#** asserts or when software sets the Secondary Bus Reset bit in the Bridge Control register. Secondary **RST#** resets the secondary bus interface of the bridge.

### *3.5.1.2. SHPC Initialization Sections*

There are five sections of logic within a bridge integrated with an SHPC with distinct initialization requirements.

1. **Primary bus interface.** These are the state machines that connect the bridge to the primary bus. For a host bridge, the primary bus is the host-side interface.

   This section of a bridge integrated with an SHPC is initialized by primary reset.

2. **Bridge configuration registers.** For a PCI-to-PCI bridge, this includes the Type 01h header defined in PCI Bridge 1.1 (which includes the Status register, Command register (including SERR Enable, Memory Enable, IO Enable, etc.), Base Address

registers, Bridge Control register, Secondary Status register, bus number registers, CacheLine size, Interrupt Line, Interrupt Pin, Latency Timer, Base and Limit range registers, etc.). For bridges that support power management, this includes the PCI Power Management Capabilities List Item, excluding the PME_En and PME_Status bits.

For a host bridge, this includes internal configuration and control registers such as address range registers for the PCI interface, and host-bus interface state machines. Configuration, control, and status bits that are part of the Wakeup Context are specifically *not* included in this group.

This section of a bridge integrated with an SHPC is initialized by:

- Primary reset
- ***D3***-to-***D0*** transition

3. **Wakeup Context.** Wakeup Context is bridge and SHPC state information and logic required to enable and assert the Wakeup Signal and to report the reason for asserting the Wakeup Signal. The specific registers that must be preserved are defined in Section 6.2.3.

   The requirements for resetting the Wakeup Context are the same as those defined in PCI PM 1.1 for resetting the PME Context. PCI PM 1.1 requires PME Context to be reset by different stimuli depending upon whether the bridge/SHPC is designed to assert **PME#** from the $D3_{cold}$ state. If the device does *not* support **PME#** generation from $D3_{cold}$, these bits are initialized by primary reset.

   As required by PCI PM 1.1, Section 3.2.4, if the device *does* support **PME#** generation from $D3_{cold}$, these bits are not affected by any reset condition and must be initialized by software during initial operating system load.

4. **SHPC.** This reset group includes the SHPC Working Register set described in Section 4.5 (including hardware-initialized registers, registers with specified default values, registers that control the frequency and mode of the secondary bus, and registers that control the states of the slots). It also includes all the SHPC command state machines, slot and indicator state machines, and slot-side interface state machines.

   This section of the SHPC is initialized by primary reset.

   See Section 3.5.1.4 for the requirements for hardware or firmware to initialize the hardware-initialized registers.

   See Section 3.5.1.3 for the requirements for hardware or firmware to initialize the secondary bus frequency and mode and the states of the hot-plug slots.

5. **Secondary bus interface.** This section includes all bridge and SHPC logic that is clocked by the secondary bus clock, including secondary interface state machine, and slot-specific **RST{n}#** logic. It must be initialized by:

   - Primary reset (indirectly through secondary **RST#**, see Section 3.1.4)
   - Secondary reset

See Section 3.1.4 for the requirements for asserting and deasserting **RST{n}#**.

Table 3-16 summarizes the initialization stimuli and the sections of the bridge integrated with an SHPC that are affected.

**Table 3-16: Initialization of a Bridge Integrated with an SHPC**

| Bridge/SHPC Sections | Stimuli | | |
|---|---|---|---|
| | Primary Reset | *D3*-to-*D0* Transition | Secondary Reset |
| Primary bus interface | yes | no | no |
| Bridge configuration registers | yes | yes | no |
| Wakeup Context | (Note) | no | no |
| SHPC | yes | no | no |
| Secondary bus interface | yes | no | yes |

Note: If the device does *not* support Wakeup Signal generation from $D3_{cold}$, these bits are initialized by Primary reset. If the device *does* support Wakeup Signal generation from $D3_{cold}$, these bits are not affected by any reset condition and must be initialized by software before using the SHPC or the Wakeup Signal of other devices that are wire-ORed with the Wakeup Signal of the SHPC.

### *3.5.1.3. Initializing the Secondary Bus*

Whenever the SHPC is initialized, the clock frequency and mode and slot and indicator states must be initialized to a state that would be appropriate if no hot-plug system software were ever loaded. A clock frequency and bus mode must be selected that is appropriate for the capability of the bus and the cards that are present. PCI HP 1.1 requires that all Attention Indicators be off and all slots be in a state that would be appropriate for loading non-hot-plug system software. In a system with an SHPC, this means that all slots with a closed MRL are enabled and their Power Indicators are turned on. All slots with an open MRL are disabled and their Power Indicators are turned off.

The secondary bus speed and mode and the slot states for a bus created by a host bridge are permitted to be initialized by firmware. Furthermore, the secondary bus speed and mode and the slot states for a bus created by a PCI-to-PCI bridge (Type 01h Configuration Space header) are permitted to be initialized by firmware if both of the following are true:

- The bridge is not subordinate to another PCI-to-PCI bridge (Type 01h header).
- There are no hot-plug slots on the bridge's primary bus, as indicated by the absence of an SHPC in the superior bridge.

The secondary bus speed and mode and slot state for all other PCI-to-PCI bridges must be initialized completely by hardware methods (not firmware).

**Implementation Note: Initializing Bridges with Platform Firmware**

Platform firmware is always executed after an event that resets a host bridge. Therefore, host bridges are always permitted to be initialized by Platform firmware.

Some Platform chipsets include special-purpose PCI-to-PCI bridges. That is, they appear to configuration software the same as any other PCI-to-PCI bridge (Type 01h Configuration Space headers); however, their primary interface is electrically something other than a real PCI bus. If such a bridge appears in the configuration hierarchy subordinate to a host bridge, and if the chipset does not implement an SHPC in that host bridge, that bridge/SHPC is permitted to be initialized by Platform firmware.

### Implementation Note: Hardware Initialization of PCI-to-PCI Bridges

Some applications of PCI-to-PCI bridges are required to be initialized without the assistance of firmware because those applications have no firmware that is permanently associated with the SHPC, or because reset conditions occur after the operating system is running and the firmware is no longer available. The following are some examples of problem cases in which the bridge is reset but no firmware would be available to reinitialize it.

- A PCI-to-PCI bridge integrated with an SHPC is installed in an expansion chassis. Even if such a bridge included an Expansion ROM, Expansion ROMs are executed only at system boot time, not after a hot-insertion operation.
- A PCI-to-PCI bridge integrated with an SHPC is installed on the system board subordinate to a bridge with a Type 01h Configuration Space header. If system software sets the Secondary Bus Reset bit in the Bridge Control registers of the superior bridge, the bus between the two bridges and the subordinate bridge would be reset. In most cases, this system software has no way to execute initialization routines in firmware.
- The primary interface of a PCI-to-PCI bridge integrated with an SHPC is connected to a bus that contains hot-plug slots (controlled by another SHPC). If system software changes the mode or speed of the primary bus (using the SHPC in the superior bridge), that bus and the subordinate bridge would be reset. In most cases, this system software has no way to execute initialization routines in firmware.

Section 3.5.1.4 requires the hardware-initialized registers of the SHPC to be initialized by an EEPROM or similar means in the same cases in which hardware-initialization of the secondary bus is required by Section 3.5.1.3. PCI-to-PCI bridges designed for general-purpose applications are encouraged to provide a mechanism for hardware initialization of the SHPC (both the hardware-initialized registers and the secondary bus) without firmware and to provide a method for this mechanism to be disabled if the bridge is installed in an application in which initialization is permitted to be done by the firmware. Providing this mechanism guarantees that the bridge can be used in any application. Providing a method for disabling the mechanism saves the expense of the EEPROM in those applications in which firmware initialization is permitted.

The bus segment frequency and mode and the slot and indicator states are initialized with the following steps each time the SHPC is initialized. The same sequence applies both in applications assisted by firmware and those done totally by hardware. Systems that implement a power budget are permitted to limit the number of slots they turn on. The method of budgeting power and the algorithm for determining which slots to leave off are beyond the scope of this specification.

1. Hardware initialized registers are programmed with their appropriate values for this hot-plug segment.
2. If the hot-plug segment is capable of operating in 66 MHz conventional mode (as indicated by a non-zero value in the Number of Slots Available (66 MHz Conv.) field of the Slots Available register), this step is required. Otherwise, this step is optional.

   Execute a Power-Only All Slots command. As each slot (with closed MRL) powers up, the SHPC records the capability of add-in cards in those slots to operate in 66 MHz conventional mode.

3. The highest secondary frequency and mode is selected that satisfies both of the following requirements:

   - The Number of Slots Available field (of the Slots Available register) for that frequency and mode (see Section 4.5.2) includes all applicable slots (as defined below). (See Section 3.2.7 concerning mixing hot-plug slot and non-hot-plug devices on the same bus.)
   - All cards in applicable slots (as defined below) support the frequency and mode.

   An applicable slot is a slot for which all of the following are true:

   - The slot's MRL is closed, as indicated by the MRL Sensor bit in the Slot Status field of the Logical Slot register (see Section 4.5.12.1).
   - The slot contains an add-in card, as indicated by that slot's **PRSNT1#** and **PRSNT2#** bits in the Slot Status field of the Logical Slot register (see Section 4.5.12.1).
   - If the system implements a power budget, the system is capable of supplying the power required by the add-in card as determined by the states of the **PRSNT1#** and **PRSNT2#** bits in the Slot Status field of the Logical Slot register (see Section 4.5.12.1).

4. If the optional step 2 above was executed, and the contents of the Slots Available register for the selected frequency is less than the contents of the Number of Slots Implemented field of the Slot Configuration register, execute a Slot Disable and Power Indicator Off command for each slot that is implemented but not available at the selected frequency and mode.
5. Execute a Set Bus Segment Speed/Mode command to change the bus speed/mode to that selected in the step 3.
6. Execute an Enable All Slots command to enable and turn on the Power Indicators for all slots that are available at the present speed/mode and have a closed MRL.

#### *3.5.1.4. Initializing the Hardware-Initialized Registers*

The default values of registers designated in Section 4.5 as "hardware initialized" are loaded from outside the SHPC. Hardware-initialized registers and secondary bus state must be self-initializing (without the use of firmware) in the same cases. That is, if an SHPC is allowed in Section 3.5.1.3 to initialize the secondary bus frequency and mode and the slot states using firmware, that SHPC is also allowed to initialize the hardware initialized registers using firmware. Conversely, if an SHPC is required in Section 3.5.1.3 to initialize the secondary bus automatically by hardware mechanisms (not firmware), that SHPC must also initialize the hardware-initialized registers automatically by hardware mechanisms. Possible hardware mechanisms include, but are not limited to, pin strapping or a serial EEPROM.

### 3.5.2. Summary of SHPC Required Inputs

All slot-specific input signals must be supported by the SHPC. All inputs are required to be debounced before they affect state machine behavior, System Interrupt, system error, or Wakeup Signal generation, or are directly or implicitly visible to software via the programming model (see Appendix A). Required SHPC slot-side interface input signals are:

- **PRSNT2{n}#**
- **PRSNT1{n}#**
- **PWRFLT{n}#**
- **MRL{n}#**
- **BUTTON{n}#**
- **M66EN{n}** (if the bridge supports 66 MHz conventional mode)
- **PCIXCAP{n}** (if the bridge supports PCI-X)

### 3.5.3. Summary of SHPC Required Outputs

The SHPC programming model requires a minimum set of output controls to be completely functional. Required Slot Control Logic outputs are:

- **PWREN{n}#**
- **BUSEN{n}#**
- **RST{n}#**
- **ATTLED{n}#**
- **PWRLED{n}#**

If a bridge with an SHPC does not support Message Signaled Interrupts, it is required to implement an **INTx#** pin. A bridge with an SHPC that supports Message Signaled Interrupts is recommended also to support an **INTx#** pin, for compatibility with systems that do not support Message Signaled Interrupts.

In addition, PCI-to-PCI bridges must implement **PME#** and an **SERR#** pin. Host bridges optionally implement **WAKE#** and must implement a means to generate a critical system interrupt, for example, NMI or machine check.

## 3.6. Summary of SHPC Platform Requirements

The section summarizes the Platform requirements stated elsewhere in this chapter.

### 3.6.1. Power Fault Detection Requirements

Independent main power fault detection is required for all hot-plug slots. As a minimum, over-current fault detection is required for all main supply voltage rails, and under voltage fault detection is required for the 3.3V and 5V main supply voltage supply rails. A power fault signal must be provided to the Slot Control Logic. Upon fault detection, the main power must be removed from the slot. The main power fault signal must be

latched (remain asserted) until the fault is explicitly cleared when the slot is turned off by software.

When the Platform supports **3.3Vaux** fault detection, auxiliary power faults must be reported through the same pin as main power faults. When a **3.3Vaux** power fault is detected, slot-specific **3.3Vaux** must be turned off. Upon **3.3Vaux** power fault detection, the fault signal must be latched (remain asserted) until the MRL is opened.

### 3.6.2. MRL Sensor Support

Slots that support **3.3Vaux** are required to include an MRL Sensor. The MRL Sensor is required to turn the **3.3Vaux** voltage on and off, independent of the **PWREN{n}** signal state.

Slots that support the MRL Switched Signals are required to use the MRL Sensor to connect these signals to the power management or system management bus infrastructure.

### 3.6.3. Power Controller Timing

The Power Controller must be designed to power on or power off a slot within the minimum command completion times (timing parameter $T_{pccc}$) specified by the SHPC.

That is, when a Slot Power Only command is issued, slot power must be valid before the Controller Busy bit is cleared. Similarly, when a Slot Disable command is issued, slot power must be completely off ($V_{off}$) before the Controller Busy bit is cleared.

# 4. SHPC Programming Interface

This chapter defines the software-programming interface for the SHPC. The programming interface assumes the system design requirements defined in Section 4.1. This chapter also describes how software discovers SHPCs in a system (see Section 4.3).

The SHPC programming interface includes the SHPC Working Register set defined in Section 4.5. The SHPC Working Register set implements the command interface defined in Section 4.6. The command interface gives software the ability to control hot-plug slots and the bus segment that contains the hot-plug slots. The SHPC monitors the slots it controls for events. When the SHPC detects an event, it may optionally be programmed to generate a System Interrupt, assert the Wakeup Signal, or pulse **SERR#** (see Section 4.7).

All SHPCs must implement the programming interface defined in this section.

## 4.1. SHPC System Design Requirements

The SHPC must be integrated into a PCI-to-PCI bridge (see Section 4.3.1) or host bridge (see Section 4.3.2). The SHPC must not be integrated with any other type of bridge.

Integration with a bridge does not require the SHPC to be the same physical part as the bridge. The SHPC may optionally be a separate physical part from the bridge as long as the other requirements of this specification are met.

A PCI-to-PCI bridge with an integrated SHPC must implement the following register sets (see Section 4.3.1):

- SHPC Capabilities List Item
- SHPC Working Register set

A host bridge with an integrated SHPC must implement the Host Bridge Register Block defined in Section 4.3.2.1. This includes the SHPC Working Register set which is contained within the Host Bridge Register Block.

The following requirements apply to Platforms that include an SHPC:

- Each SHPC controls slots only on the secondary bus of the bridge it is integrated with. Software determines the secondary bus number of a PCI-to-PCI bridge by reading the Secondary Bus Number register in the Type 01h Configuration Space header. Software determines the secondary bus number of a host bridge by reading the Bus Number register contained in the HBRB Header.
- Only one SHPC is integrated with any one particular bridge (that is, a bridge cannot contain two SHPC Capabilities List Items or two SHPC Working Register sets). This does not preclude the integration of multiple bridges into a single physical part. However, each bridge/SHPC must be logically distinct.
- Only one SHPC controls slots on any one particular PCI bus segment (that is, all hot-plug slots on a PCI bus segment must be controlled by only one SHPC).
- Device numbers of hot-plug slots controlled by a SHPC must increment by 1 starting with the First Device Number field in the Slot Configuration register (see Section 4.5.3).
- Each bus segment that includes at least one hot-plug slot must include an SHPC associated with the source bridge for that bus.

When bus loading restrictions limit the number of hot-plug slots that may be enabled, the hot-plug slots with the lowest device numbers must be used. See Section 4.5.2.

## 4.2. Register Attribute Definitions

This chapter uses the following nomenclature for the *Register Attribute* column of tables describing registers contained in the SHPC programming interface:

- R/W – *Read/Write.* Register bits of this type can be read or written. If software reads a bit with this attribute, the last value written is returned.
- RO – *Read Only.* Register bits of this type can only be read. Writes are ignored and have no effect.
- R/WC – *Read/Write Clear.* Register bits of this type can be read or written. Writing a 1 clears R/WC bits to 0. Writing a 0 to these bits is ignored.
- RsvdP – *Reserved and Preserved.* Register bits of this type are reserved for future use as R/W bits. The value read is undefined. Writes are ignored. Software must follow these rules when accessing RsvdP bits:
  - Software must ignore RsvdP bits when testing values read from these registers.
  - Software must not depend on a RsvdP bit's ability to retain information when written.
  - Software must always write back the value read in the RsvdP bits when writing one of these registers.
- RsvdZ – *Reserved and Zero.* Register bits of this type are reserved for future use as R/WC bits. The value read is undefined. Writes are ignored. Software must follow these rules when accessing RsvdZ bits:
  - Software must ignore RsvdZ bits when testing values read from these registers.
  - Software must not depend on a RsvdZ bit's ability to retain information when written.
  - Software must always write 0 to RsvdZ bits when writing one of these register.
- HwInit – Hardware Initialized. Register bits of this type must be initialized by system firmware or hardware mechanisms as described in Section 3.5.1.4. Hardware mechanisms could include such things as pin strapping or a serial EEPROM. If register bits of this type are initialized by firmware, the bits must be implemented as 'write once,' to allow firmware to write the initial value after every primary reset. (See Section 3.5.1.1.) Writes to initialize these registers must be a full DWORD. Writes of other size are not defined. After they are initialized, HwInit bits are Read Only until they are reset by primary reset.

## 4.3. SHPC Identification and Addressing

The following sections describe how software discovers an SHPC in a bridge and how the SHPC Working Register set is addressed.

### 4.3.1. SHPC Integrated with a PCI-to-PCI Bridge

A PCI-to-PCI bridge with an integrated SHPC implements the register layout shown in Figure 4-1.

**Type 01h Configuration Header (ref) And SHPC Capabilities List Item**

| Byte Offset | (Configuration Space) | | | |
|---|---|---|---|---|
| 00h | Device ID | | Vendor ID | |
| 04h | Status | | Command | |
| 08h | Class Code (0604XXh) | | | Revision ID |
| 0Ch | BIST | Header Type | Pri. Latency Timer | Cacheline Size |
| 10h | Base Address Register 0 | | | |
| 14h | Base Address Register 1 | | | |
| 18h | Sec. Latency Timer | Sub. Bus Number | Sec. Bus Number | Pri. Bus Number |
| 1Ch | Sec. Status | | I/O Lim | I/O Base |
| 20h | Memory Limit | | Memory Base | |
| 24h | Prefetchable Mem Lim | | Prefetchable Mem Base | |
| 28h | Prefetchable Base Upper 32 bits | | | |
| 2Ch | Prefetchable Base Limit Upper 32 bits | | | |
| 30h | I/O Limit Upper 16 bits | | I/O Base Upper 16 bits | |
| 34h | Reserved | | | Cap Ptr |
| 38h | Expansion ROM Base Address | | | |
| 3Ch | Bridge Control | | Interrupt Pin | Interrupt Line |
| | … | | | |
| N | DWORD Select/CIP | | Next Cap. Ptr | SHPC Cap. ID |
| N+04h | DWORD Data Register | | | |

**SHPC Working Registers**

| DWORD Select Index | (Memory Space) | |
|---|---|---|
| 00h | SHPC Base Offset Register | |
| 01h | Slots Available I Register | |
| 02h | Slots Available II Register | |
| 03h | Slot Configuration Register | |
| 04h | SHPC Prog. I/F | Secondary Bus Configuration Register |
| 05h | Controller Command Status Register | Controller Command Register |
| 06h | Interrupt Locator Register | |
| 07h | SERR Locator Register | |
| 08h | Controller SERR-INT Register | |
| 09h | 1st Logical Slot Register | |
| | … | |
| 26h | 30th Logical Slot Register | |
| 27h | 31st Logical Slot Register | |
| 28h | Vendor Specific | |
| 29h | Vendor Specific | |
| | … | |

**Figure 4-1: Register Layout in a PCI-to-PCI Bridge**

PCI-to-PCI bridges implement a Type 01h Configuration Space header (see PCI Bridge 1.1). Software determines that a PCI-to-PCI bridge integrates an SHPC by finding the SHPC Capabilities List Item in the bridge's Capabilities List. Software

determines that a PCI-to-PCI bridge supports a Capabilities List by reading the Status register contained in the bridge's Configuration Space header. If the bridge contains a Capabilities List (Capabilities List bit in the Status register set to 1), software searches the linked list of capabilities. If the SHPC Capabilities List Item is found, the bridge contains an SHPC that controls at least one hot-plug slot. Table 4-1 and Table 4-2 define the specific fields of the SHPC Capabilities List Item. The SHPC Capabilities List Item must not exist in the Capabilities List of bridges that do not contain an integrated SHPC or bridges that contain an SHPC that does not control any hot-plug slots in the Platform. The format of the SHPC Capabilities List Item is shown in Figure 4-2.

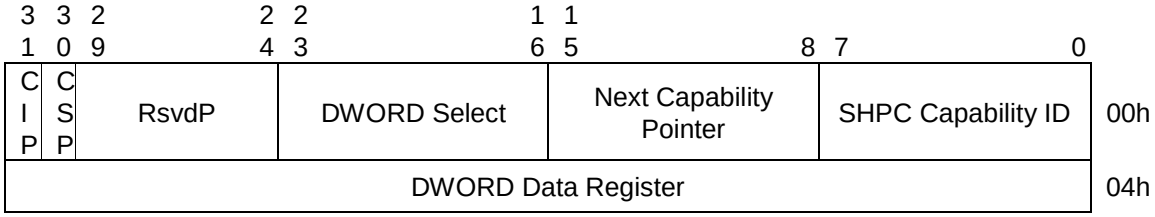



**Figure 4-2: Format of SHPC Capabilities List Item in a PCI-to-PCI Bridge**

**Table 4-1: SHPC Capabilities List Item First DWORD**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 7:0 | **SHPC Capability ID** – Used to detect the presence of an SHPC integrated with a PCI-to-PCI bridge. The SHPC Capability ID must be set to 0Ch. | RO |
| 15:08 | **Next Capability Pointer** – The offset of the next Capabilities List Item or 00h if no other items exist in the linked list of capabilities. (See PCI 2.2.) | RO |
| 16:23 | **DWORD Select** – Selects the DWORD from the SHPC Working Register set that is accessible through the DWORD Data register. Accesses to the DWORD Data register have no effect on the DWORD Select field. The value of 0 selects the first DWORD of the SHPC Working set. A value of 1 selects the second DWORD, and so on. (See Figure 4-1.)<br><br>This field has a default value of 0. | R/W |
| 30 | **Controller System Error Pending (CSP)** – This bit is set when one or more bits are set in the SERR Locator register (see Section 4.5.10). This bit is cleared when no bits are set in the SERR Locator register. | RO |
| 31 | **Controller Interrupt Pending (CIP)** – This bit is set when one or more bits are set in the Interrupt Locator register (see Section 4.5.9). This bit is cleared when no bits are set in the Interrupt Locator register. | RO |

**Table 4-2: SHPC Capabilities List Item Second DWORD**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 31:00 | **DWORD Data** – This field allows software to access the SHPC Working Register set via the Capabilities List Item in Configuration Space. The DWORD Select field selects the SHPC Working Register set DWORD that is accessed by reads and writes to this register. Accessing SHPC Working Register set registers through this field behaves the same as accessing them through memory-mapped accesses (see Section 4.5). Multiple accesses to the DWORD Data register continue to affect the same DWORD if the DWORD Select field is unchanged.<br><br>If the PCI-to-PCI bridge integrated with the SHPC is not in the ***D0*** power management state, reads from this register must complete successfully but the returned value is undefined and the behavior of writes is undefined. | R/W |

A PCI-to-PCI bridge with an integrated SHPC must provide two methods for accessing the SHPC Working Register set:

1. Memory-mapped access. The SHPC Working Register set is accessed through a range of memory addresses beginning at an address determined by adding the contents of the bridge's Base Address register to the byte offset obtained from the SHPC Base Offset register (contained in the SHPC Working Register set). See Section 4.5.1.
2. Indirect access. The SHPC Working Register set is accessed using the DWORD Select/DWORD Data fields located in the SHPC Capabilities List Item in Configuration Space.

A PCI-to-PCI bridge with integrated SHPC is required to implement a 64-bit Base Address register. This means the SHPC Working Register set is located anywhere in the 64-bit memory address space. Additionally, the SHPC must implement the SHPC Base Offset register in the SHPC Working Register set. Software uses this register to identify the offset that must be added to the value in the 64-bit Base Address register to access the SHPC Working Register set. A Base Address register that includes only the SHPC Working Register set is recommended to be marked prefetchable (as specified in PCI 2.2). A Base Address register that includes locations other than the SHPC Working Register set is marked either prefetchable or nonprefetchable, depending upon the requirements of the other locations. (In many systems the use of nonprefetchable memory is more restrictive on the system software.) If the Base Address register is marked nonprefetchable, the bridge is permitted to limit read accesses to the SHPC Working Register set to not more than a single naturally aligned DWORD per transaction, and to terminate all other read transactions with Target-Abort. As with any PCI device, the bridge must properly execute memory write transactions of any length, but is permitted to disconnect them into single data phase transactions. (Multiple-data-phase memory write transactions sometimes occur because smaller transactions were combined, as defined in PCI 2.2.)

**Implementation Note: Using the SHPC Base Offset Register to Address the SHPC Working Register Set**

Software must perform the following steps to compute the base memory mapped I/O address of the SHPC Working Register set when a PCI-to-PCI bridge integrates an SHPC:

1. Software obtains the base address from the PCI-to-PCI bridge's 64-bit Base Address register.
2. Using the DWORD Select/DWORD Data fields in the SHPC Capabilities List Items, software reads the value contained in the SHPC Base Offset Register at DWORD select index 00h.
3. To obtain the base memory mapped I/O address of the SHPC Working Register set, software adds the offset value obtained from step 2 to the Base Address register value obtained in step 1. The computed base memory address corresponds to byte 0 of the SHPC Base Offset register.

## 4.3.2. SHPC Integrated with a Host Bridge

A host bridge provides a connection path between a host bus and PCI bus for I/O, memory, and PCI configuration cycles. PCI 2.2 does not describe how host bridges are located or configured and, therefore, many valid implementations exist. These include Type 00h Configuration Space headers and memory mapped configuration registers. This lack of hardware standardization requires firmware of the machine to describe the host bridges to system software.

Operation of an SHPC that is integrated with a host bridge is through the SHPC Working Register set associated with that host bridge. To this end, a block of memory mapped registers is defined for each host bridge, the address of which is provided to system software by the firmware as described in Section 5. This block consists of a fixed header containing identification information followed by a variable size block through which a linked list of capabilities is threaded. This is designed to be analogous to the Capabilities List found in PCI Configuration Space (see PCI 2.2) and to allow additional features to be added to host bridges in the future.

A host bridge with an integrated SHPC must implement the Host Bridge Register Block as described in this section. The Capability List in this Host Bridge Register Block must include a Capability List Item for the SHPC as defined in Section 4.3.2.4. See Section 5.5.3 for additional restrictions on Host Bridge Registers Blocks in systems using ACPI.

### *4.3.2.1. Host Bridge Register Block*

The Host Bridge Register Block (HBRB) begins with an HBRB Header followed by an optional list of HBRB Capability List Items. This is illustrated by Figure 4-3.

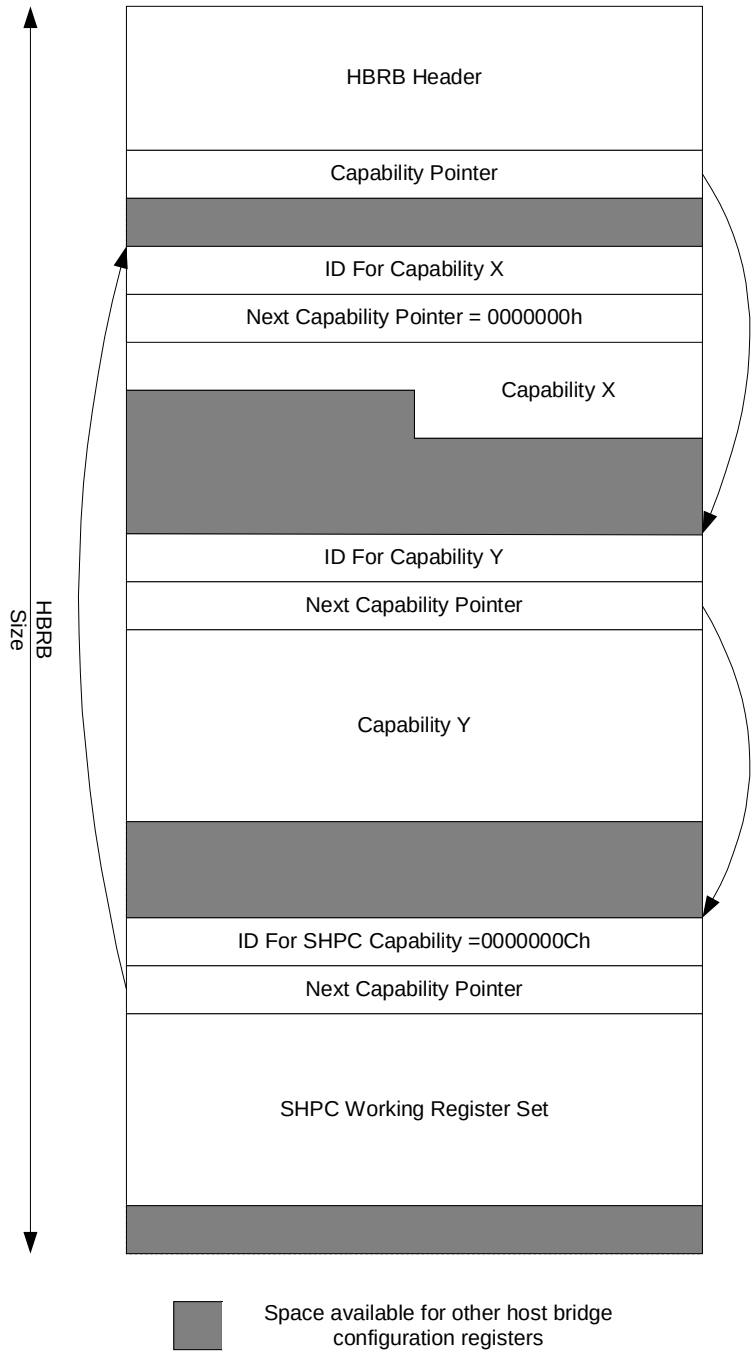



**Figure 4-3: Host Bridge Register Block**

### *4.3.2.2. HBRB Header*

The HBRB Header consists of identification data (similar to that of a Configuration Space header) and a pointer to a linked list of capabilities. The HBRB Header must be DWORD aligned.

| 31 | 24 | 23 | 16 | 15 | 8 | 7 | 0 | Byte Offset |
|---|---|---|---|---|---|---|---|---|
| Device ID | | | | Vendor ID | | | | 00h |
| HBRB Version | | Bus Number | | Prog Interface | | Revision ID | | 04h |
| Subsystem ID | | | | Subsystem Vendor ID | | | | 08h |
| HBRB Size | | | | | | | | 0Ch |
| Capabilities List Pointer | | | | | | | | 10h |

**Figure 4-4: HBRB Header**

**Table 4-3: HBRB Header**

| DWORD Offset | Bit Location | Description | Register Attribute |
|---|---|---|---|
| 00h | 15:00 | **Vendor ID** – This field identifies the manufacturer of the device.<br><br>It is subject to the same rules as a device in Configuration Space, described in PCI 2.2, Section 6.2.1.<br><br>If this host bridge also appears as a device in Configuration Space, this field must contain the same value as the register with the same name in the host bridge Configuration Space header. | RO |
| 00h | 31:16 | **Device ID** – This field identifies the particular device.<br><br>It is subject to the same rules as a device in Configuration Space, described in PCI 2.2, Section 6.2.1.<br><br>If this host bridge also appears as a device in Configuration Space, this field must contain the same value as the register with the same name in the host bridge Configuration Space header. | RO |
| 01h | 7:0 | **Revision ID** – This register specifies a device specific revision identifier.<br><br>It is subject to the same rules as a device in Configuration Space, described in PCI 2.2, Section 6.2.1.<br><br>If this host bridge also appears as a device in Configuration Space, this field must contain the same value as the register with the same name in the host bridge Configuration Space header. | RO |

| | | | |
|---|---|---|---|
| 01h | 15:08 | **Prog. Interface** – This field identifies a specific register-level programming interface.<br><br>It is subject to the same rules as for the lower byte (offset 09h) of the Class Code field of a device in Configuration Space, described in PCI 2.2, Section 6.2.1.<br><br>If this host bridge also appears as a device in Configuration Space, this field must contain the same value as the lower byte (offset 09h) of the Class Code field in the host bridge Configuration Space header. | RO |
| 01h | 23:16 | **Bus Number** – This field contains the PCI Bus Number of the bus that contains the slots controlled by this SHPC.<br><br>This field must be implemented as a read-only copy of the register that assigns the secondary bus number on host bridges integrated with an SHPC. | RO |
| 01h | 31:24 | **HBRB Version** – This field identifies the format of the HBRB Header. A value of 01h identifies the HBRB Header format defined in this specification. All other values are reserved. | RO |
| 02h | 15:00 | **Subsystem Vendor ID** – This field identifies the vendor who constructed the Platform in which this host bridge resides.<br><br>It is subject to the same rules as a device in Configuration Space, described in PCI 2.2, Section 6.2.4, with the exception that it is required (not optional).<br><br>If this host bridge also appears as a device in Configuration Space, this field must contain the same value as the register with the same name in the host bridge Configuration Space header. If the registers are designed to be initialized by firmware, a write to either location initializes both locations after primary reset. Subsequent writes to either location are ignored. | HwInit |
| 02h | 31:16 | **Subsystem ID** – This field contains a vendor specific code to identify a particular vendor's implementation of a hot plug segment. It is recommended that it be updated to allow the vendor to uniquely identify a system design if major components (for example, power controller) are changed.<br><br>It is subject to the same rules as a device in Configuration Space, described in PCI 2.2, Section 6.2.4, with the exception that it is required (not optional).<br><br>If this host bridge also appears as a device in Configuration Space, this field must contain the same value as the register with the same name in the host bridge Configuration Space header. If the registers are designed to be initialized by firmware, a write to either location initializes both locations after primary reset. Subsequent writes to either location are ignored. | HwInit |

| | | | |
|---|---|---|---|
| 03h | 31:00 | **HBRB Size** – This field contains the size in bytes of the Host Bridge Register Block from the start of the HBRB Header to the last byte decoded by the host bridge. This must include all HBRB Capabilities List Items and any vendor specific host bridge registers. The value in this field must be an integer multiple of 4 bytes. That is, it must represent an integer number of DWORDS. | RO |
| 04h | 31:00 | **Capabilities List Pointer** – This field contains the offset in bytes from the start of the HBRB Header to the first HBRB Capabilities List Item. If the HBRB does not contain any Capabilities List Items, this field must be 0000 0000h. The value in this field must be an integer multiple of 4 bytes. That is, it must represent an integer number of DWORDS. The bottom two bits are reserved and must be implemented as 00b. However, software must mask them to allow for future uses of these bits. | RO |

### *4.3.2.3. HBRB Capabilities List Items*

Each HBRB Capabilities List Item consists of a 32-bit ID field (assigned by the PCI SIG), a 32-bit pointer in the Host Bridge Register Block to the next capability, and some additional registers immediately following the pointer to implement that capability. Each capability must be DWORD aligned. The bottom two bits of all pointers (including the initial pointer at offset 10h in the HBRB Header) are reserved and must be implemented as 00b. However, software must mask them to allow for future uses of these bits.

HBRB Capabilities List Items are permitted to appear in any order in the Host Bridge Register Block. That is, an item that is later in the list is permitted to appear either closer or farther from the HBRB Header than the previous items in the list.

A pointer value of 0000 0000h is used to indicate the last capability in the list. Figure 4-3 shows how this list is constructed.

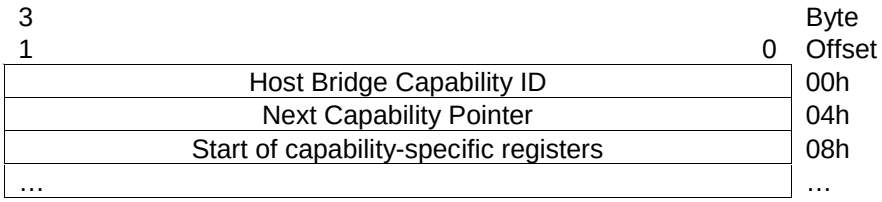



**Figure 4-5: HBRB Capability List Item**

Each defined capability must have a Host Bridge Capability ID assigned by the PCI SIG. These IDs are assigned and handled much like the Class Codes defined in PCI 2.2. Each capability must define the detailed register map for that capability.

**Table 4-4: HBRB Capability List Item**

| DWORD Offset | Bit Location | Description | Register Attribute |
|---|---|---|---|
| 00h | 31:00 | **Host Bridge Capability ID** – This field contains an ID assigned by the PCI SIG. | RO |
| 01h | 31:00 | **Next Capability Pointer** – This field contains the offset from the start of the Host Bridge Register Block of the next HBRB Capabilities List Item in the Capabilities List. A value of 0000 0000h terminates the list. | RO |

#### 4.3.2.4. The SHPC HBRB Capabilities List Item

The SHPC HBRB Capabilities List Item has a Host Bridge Capability ID of 0000 000Ch. The SHPC Working Register set directly follows the Next Capability Pointer as illustrated by Figure 4-6. If this host bridge also appears as a device in Configuration Space, the Capabilities List in Configuration Space must *not* include an SHPC Capabilities List Item. The Capabilities List Item must *not* exist in the Capabilities List of bridges that do not contain an integrated SHPC or bridges that contain an SHPC that does not control any hot-plug slots in the Platform.

| DWORD Offset | | | |
|---|---|---|---|
| 00h | Host Bridge Capability ID = 0000 000Ch | | |
| 01h | Next Capability Pointer | | |
| 02h | SHPC Base Offset Register | | |
| 03h | Slots Available I Register | | |
| 04h | Slots Available II Register | | |
| 05h | Slot Configuration Register | | |
| 06h | SHPC Prog. I/F | SHPC MSI Control Register | Secondary Bus Configuration Register |
| 07h | Controller Command Status Register | Controller Command Register | |
| 08h | Interrupt Locator Register | | |
| 09h | SERR Locator Register | | |
| 0Ah | Controller SERR-INT Register | | |
| 0Bh | 1st Logical Slot Register | | |
| | … | | |
| 28h | 30th Logical Slot Register | | |
| 29h | 31st Logical Slot Register | | |

**Figure 4-6: SHPC HBRB Capabilities List Item**

## 4.4. SHPC Slot Identifiers

Each slot controlled by an SHPC has the following numerical identifiers associated with it:

- Logical Slot Number – this identifier is an index into the set of Logical Slot registers.
- PCI Device Number – this identifier is used for PCI configuration access addressing of devices in a slot.
- Physical Slot Number – this required identifier is used to indicate a slot to the user and is usually labeled externally on the Platform chassis. See Section 2.2.7 for restrictions on the physical slot numbers.

Table 4-5 shows the starting value of each identifier and the direction of enumeration. All three identifiers increment or decrement by 1 starting with the "Starting Value" listed in Table 4-5.

**Table 4-5: Slot Identifiers**

| Slot Identifier | Starting Value | Direction of Enumeration |
|---|---|---|
| Logical Slot Number | 1 | Up |
| PCI Device ID | First Device Number field of Slot Configuration register | Up |
| Physical Slot Number | Physical Slot Number field of Slot Configuration register | Up or Down depending on the value in the Physical Slot Number Up/Down field of the Slot Configuration register |

Table 4-5 is used to translate between identifiers. For example, a hot-plug slot with device number equal to the contents of the First Device Number field corresponds to Logical Slot Number 1. A hot-plug slot with device number equal to the contents of the First Device Number field plus 1 corresponds to Logical Slot Number 2, and so on.

## Implementation Note: SHPC Slot Identifier Example

Figure 4-7 shows an example of the relationships among the slot identifiers and how those identifiers relate to the slots controlled by an SHPC. In this example, the Slot Configuration register would contain the following values (see Section 4.5.3):

| Register Field | Value |
|---|---|
| Number of Slots Implemented | 4 (Note) |
| First Device Number | 5 |
| Physical Slot Number Up/Down | 0 (down) |
| Physical Slot Number | 4 |

Note: The SHPC may optionally be designed to control more slots than are connected to it in a system implementation. For example, in this implementation, the SHPC could be designed to control eight slots even though it is currently controlling only four slots in the system.

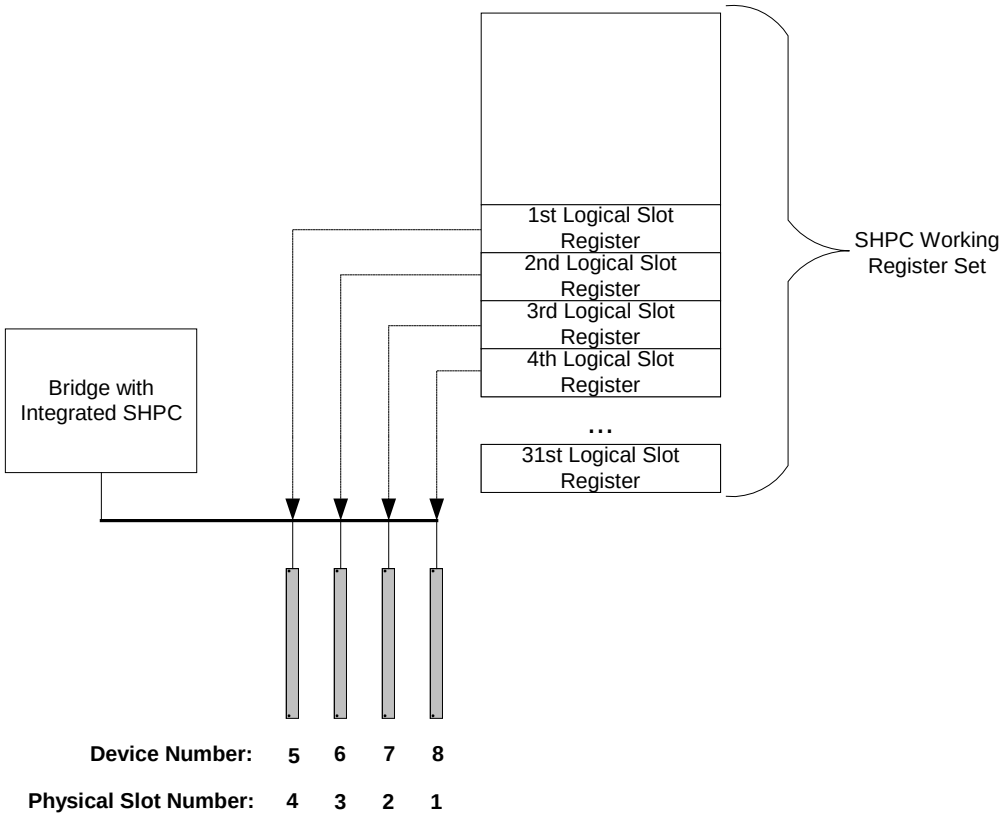



**Figure 4-7: SHPC Slot Identifier Example**

The first slot controlled by the SHPC in this example has a device number of 5 and a physical slot number of 4. Physical slot numbers in this example start with 4 and count down to 1 because the Physical Slot Number Up/Down field is 0. Device numbers start with 5 and count up to 8. Device numbers must always count up by 1. The first Logical Slot register in the SHPC Working Register set always corresponds to the first controlled slot, the second Logical Slot register corresponds to the second controlled slot, and so on.

Software uses the logical slot number in the Target Slot field of the Controller Command register to select a slot when a Slot Operation command is issued to the SHPC. In this example, software would set Target Slot to 3 to issue a command to the third controlled slot. (The third controlled slot corresponds to Logical slot-3/Device Number-7/Physical Slot Number-2.) See Section 4.6 for information on controller command support.

## 4.5. SHPC Working Register Set

Software uses the SHPC Working Register set for the following tasks:

- Query SHPC configuration information
- Control hot-plug slots and associated indicators
- Discover and service events generated by the controller

When accessing these registers, there is no requirement to read or write whole DWORDs (except during initialization of HwInit registers as defined in Section 4.2). Any combination of byte accesses is allowed including zero byte accesses.

The SHPC Working Register set includes the base registers (registers from DWORD index 00h through 08h) and room to support 31 Logical Slot registers (see Figure 4-8). Each Logical Slot register corresponds to a slot controlled by the SHPC. The first Logical Slot register is associated with the first hot-plug slot controlled by the SHPC. See Section 4.4 for information on how software correlates Logical Slot registers to their corresponding physical slot.

The SHPC is permitted to support any number of hot-plug slots from 1 to 31, inclusive. The SHPC provides a Logical Slot register and associated logic for each slot it is designed to support. The Platform in which the SHPC is installed is permitted to implement any number of hot-plug slots between 1 and the number of slots the SHPC was designed to support, inclusive. The Number of Slots Implemented field in the Slot Configuration register tells software the number of slots physically connected to the SHPC and also the number of valid Logical Slot registers at the end of the Working Register set. For example, an SHPC designed to support six slots could be installed in a Platform to control three slots. In this case, the SHPC's Number of Slot Implemented field must be set to 3 (this is the number of actual slots the SHPC is controlling, not the maximum number it was designed to control).

Accessing Logical Slot registers (either using the DWORD Data/Select registers or memory-mapped accesses) that correspond to slots that are not connected to the SHPC is unspecified. As such, the behavior of writes is undefined and the value returned when read is undefined. Software must ignore any values returned from these Logical Slot registers.

Vendor specific registers are only allowed to exist beyond the 31$^{st}$ Logical Slot register (byte offset 9Fh). Proper SHPC operation must not require software to access any vendor specific registers.

| 31 … 0 | Byte Offset | DWORD Select Index |
|---|---|---|
| SHPC Base Offset | 00h | 00h |
| Slots Available I | 04h | 01h |
| Slots Available II | 08h | 02h |
| Slot Configuration | 0Ch | 03h |
| SHPC Programming Interface / SHPC MSI Control / Secondary Bus Configuration | 10h | 04h |
| Controller Command Status / Controller Command | 14h | 05h |
| Interrupt Locator | 18h | 06h |
| SERR Locator | 1Ch | 07h |
| Controller SERR-INT Enable | 20h | 08h |
| 1st Logical Slot Register | 24h | 09h |
| … | | |
| 30th Logical Slot Register | 98h | 26h |
| 31st Logical Slot Register | 9Ch | 27h |
| Vendor Specific | A0h | 28h |
| Vendor Specific | A4h | 29h |
| … | | |

**Figure 4-8: SHPC Working Register Set**

The following sections provide details on the format and usage of each register.

## 4.5.1. SHPC Base Offset Register

The SHPC Base Offset register is used by software (in conjunction with the Base Address register) to determine the memory base address of the SHPC Working Register set when an SHPC is integrated into a PCI-to-PCI bridge (see Section 4.3.1). It is not used in a host bridge. Note that the SHPC Base Offset register must be accessed initially via Configuration Space.

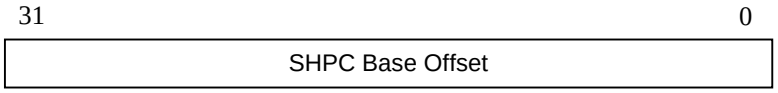



**Figure 4-9: SHPC Base Offset Register**

**Table 4-6: SHPC Base Offset Register**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 31:00 | **SHPC Base Offset** – This field contains the byte offset that must be added to the contents of the 64-bit Base Address register in a PCI-to-PCI bridge to access the SHPC Working Register set using memory-mapped accesses. In a host bridge, this register is reserved and returns 0. | RO |

## 4.5.2. Slots Available Registers

The SHPC contains five Number of Slots Available fields organized into two Slots Available registers. Each Number of Slots Available field specifies the maximum number of hot-plug slots that are permitted to be enabled at the given speed and mode on the bus segment associated with this SHPC (see Figure 4-10). This gives the system a way to enforce bus-loading restrictions for different bus speed/mode combinations. For example, a value of 3 in the Number of Slots Available (100 MHz PCI-X) field limits the maximum number of slots that can be enabled on this bus segment to three, when the bus is operating in 100 MHz PCI-X mode. The same system would be permitted to set the Number of Slots Available (66 MHz PCI-X) to 4 indicating a maximum of four slots could be enabled at 66 MHz PCI-X mode. If a bus segment does not support any hot-plug slots at a given speed/mode, the corresponding Number of Slots Available field must be 0.

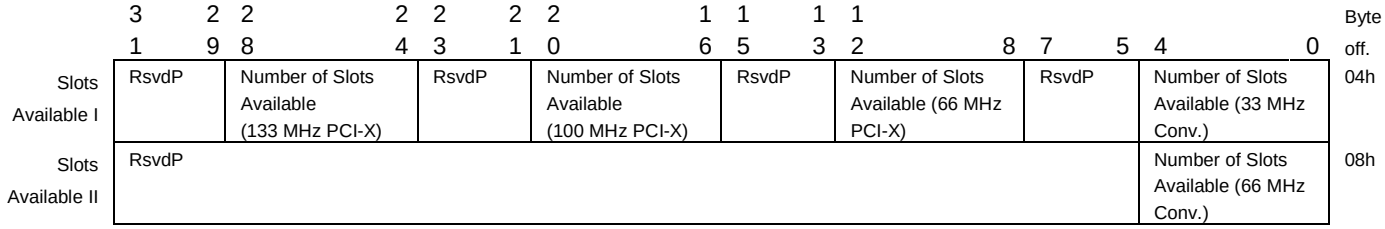



**Figure 4-10: Slots Available Registers (I and II)**

**Table 4-7: Slots Available Registers I**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 4:0 | **Number of Slots Available (33 MHz Conv.)** – Maximum number of hot-plug slots available to be enabled when the bus is running at 33 MHz conventional mode | HwInit |
| 12:08 | **Number of Slots Available (66 MHz PCI-X)** – Maximum number of hot-plug slots available to be enabled when the bus is running at 66 MHz PCI-X mode | HwInit |
| 20:16 | **Number of Slots Available (100 MHz PCI-X)** – Maximum number of hot-plug slots available to be enabled when the bus is running at 100 MHz PCI-X mode | HwInit |
| 28:24 | **Number of Slots Available (133 MHz PCI-X)** – Maximum number of hot-plug slots available to be enabled when the bus is running at 133 MHz PCI-X mode | HwInit |

**Table 4-8: Slots Available Registers II**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 4:0 | **Number of Slots Available (66 MHz Conv.)** – Maximum number of hot-plug slots available to be enabled when the bus is running at 66 MHz conventional mode | HwInit |

Slots available at a particular bus speed/mode include only the first *n* slots controlled by the SHPC, where *n* is the value read from the Number of Slots Available field for that particular speed/mode. For example, assume a system supports four total slots and the slots start at device number 5. The Number of Slots Implemented field (see Section 4.5.3) would be set to 4 and the First Device Number field would be set to 5. Assume also that the bus segment supports two enabled slots at 100 MHz PCI-X mode ('Number of Slots Available (100 MHz PCI-X)' would be 2). The slots available at 100 MHZ PCI-X mode would include the slots with device numbers 5 and 6. Therefore, the SHPC must allow software to enable slots corresponding to device numbers 5 and 6, only. If software attempts to enable a slot corresponding to any other device number, the SHPC must cause the command to fail (see Section 4.6.1). Note that this places system design requirements on the placement of slots and their associated device numbers on a particular bus segment (see Section 4.1).

The value in the Number of Slots Available field for a particular bus speed/mode must take into account the effect of non-hot-plug devices on the bus segment (both devices in non-hot-plug slots and embedded devices) as it relates to bus loading. See Section 3.2.7 for information on how systems support bus segments that contain both hot-plug and non-hot-plug devices.

### 4.5.3. Slot Configuration Register

The Slot Configuration register describes the configuration of the slots controlled by the SHPC.

The Number of Slots Implemented field affects some of the commands executed by the SHPC (see Section 4.6). The rest of this register is only used to convey Platform configuration information to the system software used to control the SHPC and has no effect on the operation of the SHPC.

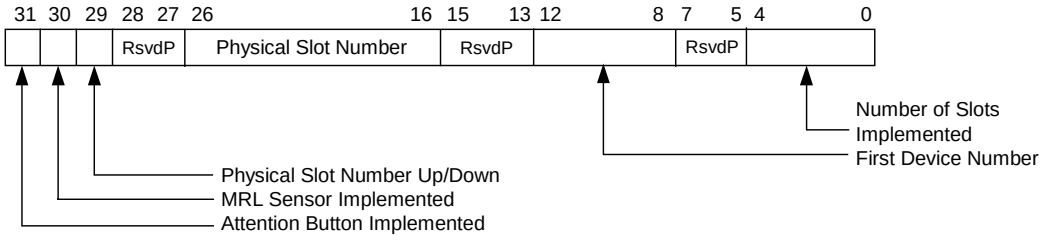



**Figure 4-11: Slot Configuration Register**

**Table 4-9: Slot Configuration Register**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 4:0 | **Number of Slots Implemented** – This field contains the number of hot-plug slots connected to the SHPC (that is, the number of slots controlled by the SHPC).<br><br>This field must not return a value of 0. (If the controller does not control any slots in the system, the SHPC Capabilities List Item must not appear in the Capabilities List.) | HwInit |
| 12:08 | **First Device Number** – This field contains the device number assigned to the first hot-plug slot on this bus segment. See Section 4.4 for the use of this register. | HwInit |
| 26:16 | **Physical Slot Number** – This field specifies the physical slot number of the device addressed by the First Device Number. This field must be hardware initialized to a value that assigns all slots (controlled by this SHPC) a slot number that is globally unique within the chassis. | HwInit |
| 29 | **Physical Slot Number Up/Down** – This bit specifies the direction of enumeration of external slot labels, beginning with the value in the Physical Slot Number field of the Slot Configuration register.<br><br>If this bit is set, each external slot label increments by 1 from the value in the Physical Slot Number field. If this bit is cleared, each external slot label decrements by 1 from the value in the Physical Slot Number field. | HwInit |
| 30 | **MRL Sensor Implemented** – This bit specifies whether MRL Sensors are implemented on the hot-plug slots controlled by the SHPC.<br><br>If this bit is set, the Platform provides an MRL Sensor for each slot controlled by this SHPC.<br><br>See Section 2.2.3 for Standard Usage Model impact and considerations related to MRL sensor implementations. Also, see Section 3.1.6 for input strapping information. | HwInit |
| 31 | **Attention Button Implemented** – This bit specifies whether the hot-plug slots controlled by this SHPC implement the optional Attention Button. If this bit is set, Attention Buttons are implemented on every slot controlled by this SHPC. | HwInit |

### 4.5.4. Secondary Bus Configuration Register

The Secondary Bus Configuration register describes the configuration of the secondary bus segment that contains the hot-plug slots controlled by the SHPC.

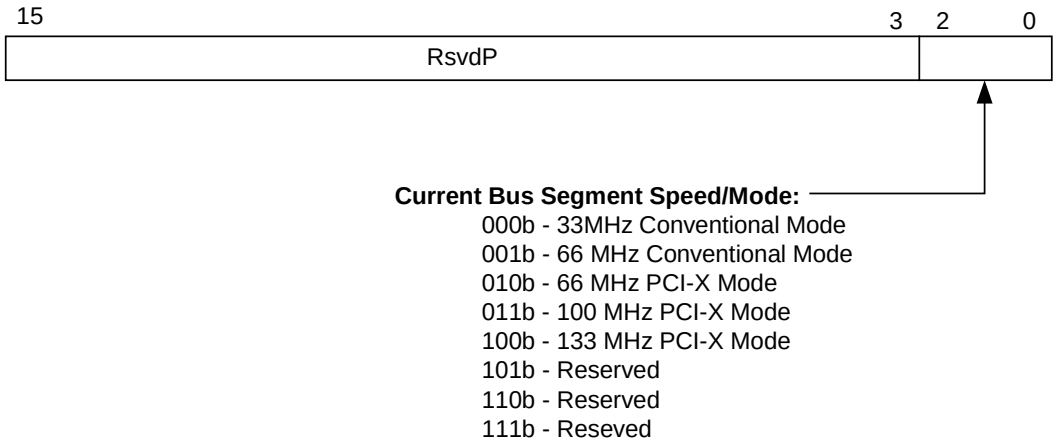



**Figure 4-12: Secondary Bus Configuration Register**

**Table 4-10: Secondary Bus Configuration Register**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 2:0 | **Current Bus Segment Speed/Mode** – Indicates the current speed and mode at which the bus segment is operating. | RO |

### 4.5.5. SHPC MSI Control Register

The SHPC MSI Control register indicates the specific message number that will be used by the SHPC to signal an interrupt when using Message Signaled Interrupts. See PCI 2.2, Section 6.8, for a description of Message Signaled Interrupts. If a bridge integrated with an SHPC implements Message Signaled Interrupts, the SHPC MSI Control register is required. If Message Signaled Interrupts are not implemented, this register is reserved and returns 0 when read. See Section 4.7.3.1 for more information on Message Signaled Interrupts.

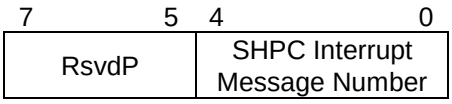



**Figure 4-13: SHPC MSI Control Register**

**Table 4-11: SHPC MSI Control Register**

| Bits Location | Description | Register Attribute |
|---|---|---|
| 4:0 | **SHPC Interrupt Message Number** – If the bridge is allocated only a single interrupt message (as determined by the Multiple Message Enable field of the Message Control register, see PCI 2.2, Section 6.8.1.3, or host bridge equivalent), this register returns 0 when read, and the 16-bit SHPC interrupt message is determined exclusively by the Message Data register defined in PCI 2.2. If the bridge is allocated more than one interrupt message, the upper bits of the 16-bit SHPC interrupt message are determined by the Message Data register, and the lower bits are determined by the value in the SHPC Interrupt Message Number field, as follows: | RO |

| SHPC Interrupt Message Number | Number of messages allocated 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| 00h | 0b | 00b | 000b | 0000b | 00000b |
| 01h | 1b | 01b | 001b | 0001b | 00001b |
| 02h | - | 10b | 010b | 0010b | 00010b |
| 03h | - | 11b | 011b | 0011b | 00011b |
| 04h | - | - | 100b | 0100b | 00100b |
| 05h | - | - | 101b | 0101b | 00101b |
| 06h | - | - | 110b | 0110b | 00110b |
| 07h | - | - | 111b | 0111b | 00111b |
| 08h | - | - | - | 1000b | 01000b |
| 09h | - | - | - | 1001b | 01001b |
| 0Ah | - | - | - | 1010b | 01010b |
| 0Bh | - | - | - | 1011b | 01011b |
| 0Ch | - | - | - | 1100b | 01100b |
| 0Dh | - | - | - | 1101b | 01101b |
| 0Eh | - | - | - | 1110b | 01110b |
| 0Fh | - | - | - | 1111b | 01111b |
| 10h | - | - | - | - | 10000b |
| 11h | - | - | - | - | 10001b |
| 12h | - | - | - | - | 10010b |
| 13h | - | - | - | - | 10011b |
| 14h | - | - | - | - | 10100b |
| 15h | - | - | - | - | 10101b |
| 16h | - | - | - | - | 10110b |
| 17h | - | - | - | - | 10111b |
| 18h | - | - | - | - | 11000b |
| 19h | - | - | - | - | 11001b |
| 1Ah | - | - | - | - | 11010b |
| 1Bh | - | - | - | - | 11011b |
| 1Ch | - | - | - | - | 11100b |
| 1Dh | - | - | - | - | 11101b |
| 1Eh | - | - | - | - | 11110b |
| 1Fh | - | - | - | - | 11111b |

### 4.5.6. SHPC Programming Interface Register

The SHPC Programming Interface identifies the format of the SHPC working registers.

**Table 4-12: SHPC Programming Interface Register**

| Bits Location | Description | Register Attribute |
|---|---|---|
| 7:0 | **SHPC Programming Interface** – Identifies the format of the SHPC Working Register set. A value of 01h identifies the SHPC Working Register set format defined in this specification. All other values are reserved. | RO |

### 4.5.7. Controller Command Register

The Controller Command register is used to issue commands to the SHPC. See Section 4.6 for a complete description of issuing commands to the SHPC.

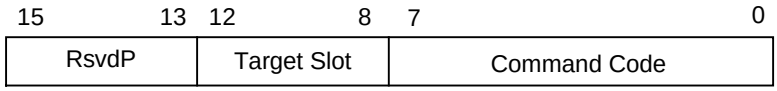



**Figure 4-14: Controller Command Register**

**Table 4-13: Controller Command Register**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 7:0 | **Command Code** – Command to be executed by the SHPC. Writing to this field triggers the SHPC to begin executing the command (see Section 4.6 for command encodings).<br><br>When read, this field returns the command code that was last written to it, even after the command has completed.<br><br>This field has a default value of 0. | R/W |
| 12:08 | **Target Slot** – This field selects the target slot for a Slot Operation command. For example, writing a 2 to this field would select the second slot for the Slot Operation command.<br><br>Software is permitted to write the Command Code and Target Slot fields simultaneously. However, software is not required to write these fields simultaneously. If the fields are not written simultaneously, the Slot Operation command targets the slot associated with the current value in this register.<br><br>If the command is not a Slot Operation command, this field is ignored.<br><br>When this field is read, it returns the value that was last written to it, even after the command has completed.<br><br>Vendor specific commands are allowed to use this field in any way.<br><br>This field has a default value of 0. | R/W |

## 4.5.8. Controller Command Status Register

The Controller Command Status register is used to determine when a command issued to the SHPC has completed, and whether the command failed or was successful. Errors not related to SHPC command execution (for example, Power Faults and Arbiter Timeouts) do not cause bits in this register to be set (see Section 4.7 for information on Power Fault and Arbiter Timeout event processing).

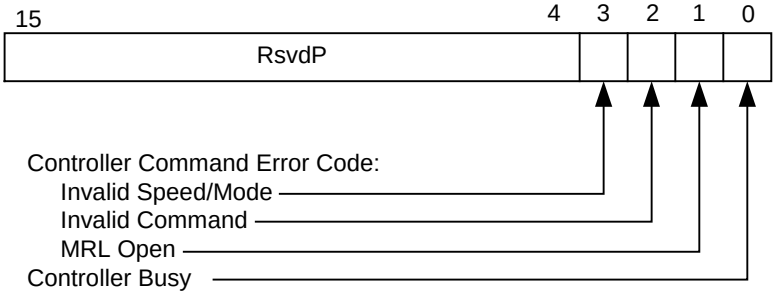



**Figure 4-15: Controller Command Status Register**

**Table 4-14: Controller Command Status Register**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 0 | **Controller Busy** – This bit changes from 0 to 1 when a command code is written to the Controller Command register. It stays set until the SHPC has completed executing the command. The SHPC ignores writes to the Controller Command register while this bit is set. This bit changes from 1 to 0 when the SHPC finishes executing a command.<br><br>The SHPC must not set this bit for any other reason. For example, this bit must not be set to 1 when the SHPC automatically powers down the slot in response to detecting a MRL open event.<br><br>See Section 4.6 for a complete description of issuing commands to the SHPC.<br><br>This bit has a default value of 0. | RO |
| 3:1 | **Controller Command Error Code** – This field shows the result of the last command completed by the SHPC. This field is updated when the Controller Busy bit transitions from 1 to 0 (indicating a command completion). If the command failed, the appropriate bit is set. If none of the bits in this field are set, the command completed successfully.<br><br>See Section 4.6 for a complete description of issuing commands and possible error conditions.<br><br>This field has a default value of 0. | RO |

### 4.5.9. Interrupt Locator Register

The Interrupt Locator register provides a way for software to quickly determine the source of interrupts (see Section 4.7.3.1 for more details).

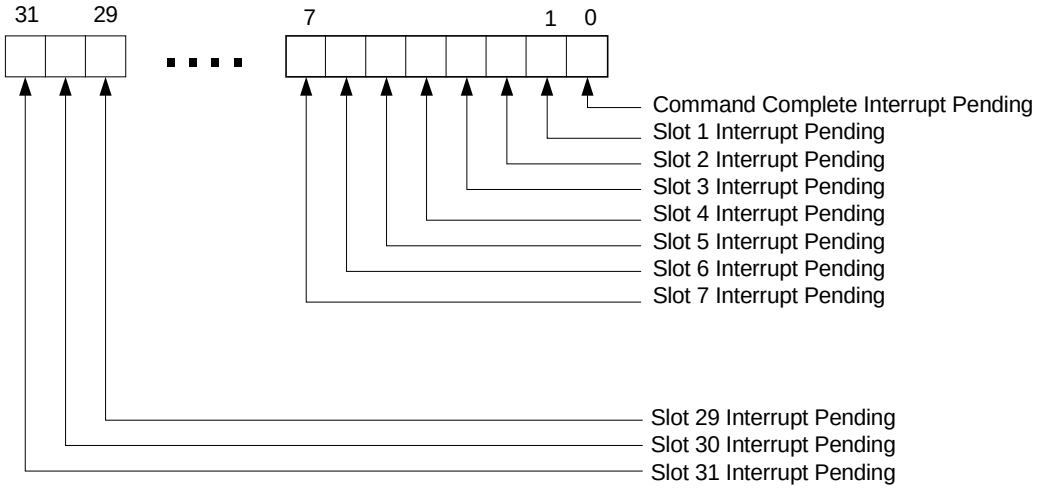



**Figure 4-16: Interrupt Locator Register**

**Table 4-15: Interrupt Locator Register**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 0 | **Command Complete Interrupt Pending** – The state of this bit is 1 when the Command Completion Detected bit is set (indicating a command completion–see Section 4.6) and the Command Complete Interrupt Mask bit located in the Controller SERR-INT register is cleared. | RO |
| 31:01 | **Slot n Interrupt Pending Bits** – A set bit in this field indicates an interrupt pending condition on the associated slot. An interrupt pending condition occurs when the SHPC detects a Slot Event, and the event's interrupt mask bit in the Slot SERR-INT Mask field is cleared. Multiple bits are set if multiple slots have an interrupt pending.<br><br>Clearing or masking all bits in the Slot Event Latch field of the slot's Logical Slot register clears that slot's bit in this field (see Section 4.5.12.2). | RO |

## 4.5.10. SERR Locator Register

The SERR Locator register provides a way for software to quickly determine the source of a System Error (see Section 4.7.3.2 for more details).

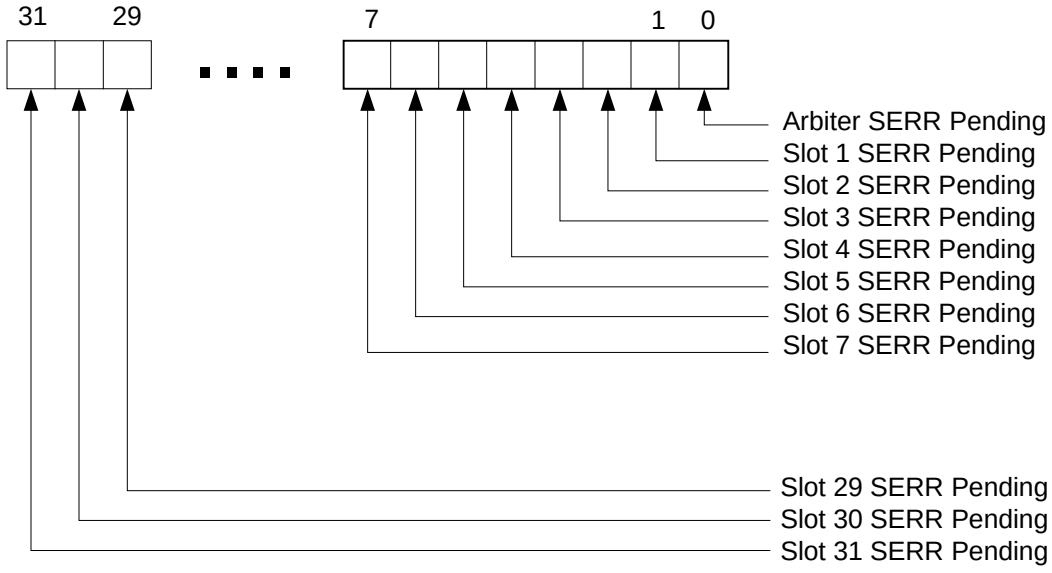



**Figure 4-17: SERR Locator Register**

**Table 4-16: SERR Locator Register**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 0 | **Arbiter SERR Pending** – The state of this bit is 1 when the Arbiter Timeout Detected bit in the Controller SERR-INT register is set and the Arbiter SERR Mask bit is cleared (see Section 3.4.1 for details on arbiter timeouts). | RO |
| 31:01 | **Slot n SERR Pending** – A set bit in this field indicates an SERR pending condition on the associated slot. An SERR pending condition occurs when the SHPC detects a slot event capable of generating an SERR and that event's SERR Mask bit in the Slot SERR-INT Mask field is cleared. Multiple bits are set if multiple slots have an SERR pending.<br><br>Clearing or masking all bits in the slot's Slot Event Latch field that are capable of generating an SERR clears that slot's bit in this field (see Section 4.5.12.2). | RO |

## 4.5.11. Controller SERR-INT Register

The Controller SERR-INT register enables and disables SERR and System Interrupt generation and reports global controller events.

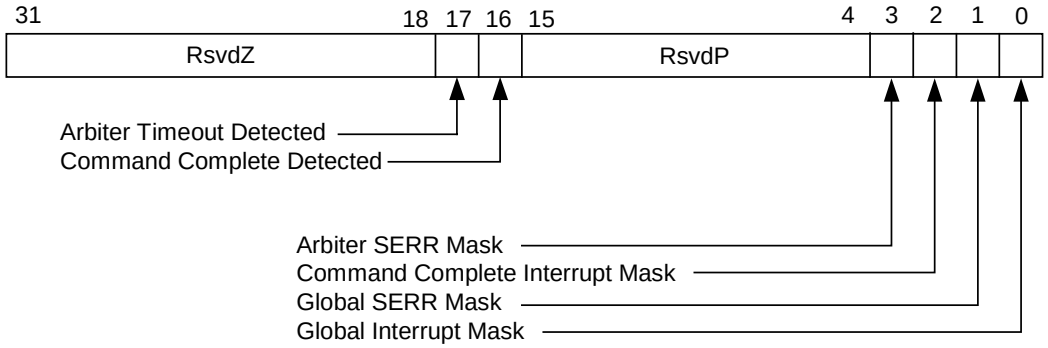



**Figure 4-18: Controller SERR-INT Register**

**Table 4-17: Controller SERR-INT Register**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 0 | **Global Interrupt Mask** – When this bit is set, System Interrupt generation by the SHPC is masked (see Section 4.7).<br><br>This bit is a mask and does not affect any bits in the Interrupt Locator register.<br><br>This bit has no effect on whether the Wakeup Signal is asserted.<br><br>This bit has a default value of 1. | R/W |
| 1 | **Global SERR Mask** – When this bit is set, SERR generation from the SHPC is masked (see Section 4.7).<br><br>This bit has a default value of 1. | R/W |
| 2 | **Command Complete Interrupt Mask** – When this bit is set, Command Completion Interrupts are masked (see Section 4.7).<br><br>This bit is a mask and does not affect whether the Command Completion Detected bit is set.<br><br>This bit has a default value of 1. | R/W |

| | | |
|---|---|---|
| 3 | **Arbiter SERR Mask** – When this bit is set, arbiter timeout SERRs are masked (see Section 4.7 for the use of this bit and Section 3.4.1 for details on arbiter timeouts).<br><br>This bit is a mask and does not affect whether the Arbiter Timeout Detected bit is set.<br><br>This bit has a default value of 1. | R/W |
| 16 | **Command Completion Detected** – This bit is set when the Controller Busy bit in the Controller Command Status register transitions from 1 to 0 (indicating a command completion).<br><br>See Section 4.6 for more details on SHPC commands.<br><br>This bit has a default value of 0. | R/WC |
| 17 | **Arbiter Timeout Detected** – This bit is set when the SHPC detects an arbiter timeout (see Section 3.4.1 for details on arbiter timeouts).<br><br>This bit has a default value of 0. | R/WC |

### 4.5.12. Logical Slot Register

Software uses the Logical Slot register to:

- Get current status on the slot
- Configure System Interrupts and System Errors generated from the slot
- Detect pending events on the slot

The SHPC must provide one Logical Slot register for each slot it controls on a bus segment.

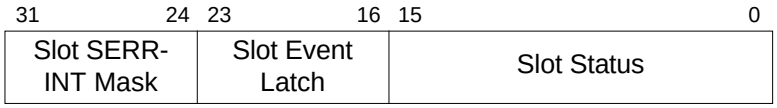



**Figure 4-19: Logical Slot Register**

### 4.5.12.1. Slot Status Field

The Slot Status field provides status information about a slot controlled by the SHPC.

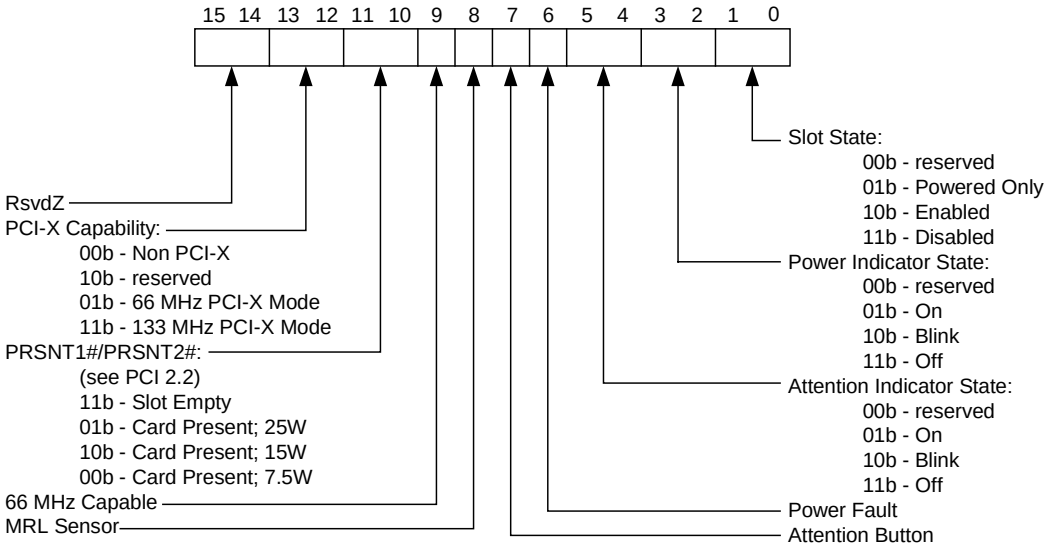



**Figure 4-20: Slot Status Field**

**Table 4-18: Slot Status Field**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 1:0 | **Slot State** – This field reports the current state of the slot (see Figure 4-20).<br><br>This field is not affected by the detection of a slot power fault by the power controller circuitry (see Section 3.1.2.2).<br><br>See Section 3.5.1.3 for information on the state of slots after coming out of reset. | RO |
| 3:2 | **Power Indicator State** – This field reports the current state of the Power Indicator associated with the slot (see Figure 4-20).<br><br>See Section 3.5.1.3 for information on the state of Power Indicators after coming out of reset. | RO |
| 5:4 | **Attention Indicator State** – This field reports the current state of the Attention Indicator associated with the slot (see Figure 4-20).<br><br>See Section 3.5.1.3 for information on the state of Attention Indicators after coming out of reset. | RO |

| | | |
|---|---|---|
| 6 | **Power Fault** – This bit reports the current state of the power fault latch in the power controller circuitry for this slot. If this bit is 1, a power fault (either isolated or connected) has been detected by the power controller circuitry.<br><br>See Section 3.1.2.3 for a description of power faults and how to clear the power fault latch in the power controller circuitry. See Section 4.7 for information about software handling of power faults. | RO |
| 7 | **Attention Button** – This bit reports the current state of the debounced Attention Button input signal for this slot (see Appendix A for a description of debounce). If this bit is 1, the Attention Button is being pressed. If this bit is 0, the Attention Button is not being pressed. | RO |
| 8 | **MRL Sensor** – This bit reports the current state of the MRL as reported by the debounced MRL Sensor input signal (see Appendix A for a description of debounce). If this bit is 1, the MRL Sensor is reporting that the MRL is open. If this bit is 0, the MRL Sensor is reporting that the MRL is closed. | RO |
| 9 | **66 MHz Capable** – This bit reports whether the add-in card is capable of running at 66 MHz conventional mode. This bit is latched as the slot is powered up or enabled, regardless of the current speed/mode of the bus (see Section 3.1.5).<br><br>If this bit is 1, the card is capable of running at 66 MHz conventional mode. If this bit is 0, the card is only capable of 33 MHz conventional mode operation.<br><br>This bit is valid only when the slot is occupied and powered or enabled. | RO |
| 11:10 | **PRSNT1#/PRSNT2#** – These bits report the current debounced state of the **PRSNT1#** and **PRSNT2#** pins on the slot (see Appendix A for a description of debounce). These bits are valid regardless of the state of the slot or speed/mode of the bus.<br><br>See Figure 4-20 for possible values. | RO |
| 13:12 | **PCI-X Capability** – These bits report the current PCI-X capability of the add-in card installed in the slot (see Figure 4-20). These bits are not valid if the slot is empty. If the slot is occupied, these bits are valid regardless of the state of the slot or speed/mode of the bus. | RO |

### *4.5.12.2. Slot Event Latch Field*

The Slot Event Latch field reports all latched events detected by the SHPC. See Section 4.7 for a description of SHPC event processing. All bits in the Slot Event Latch field are part of the Wakeup Context (see Sections 6.2.3 and 3.5.1.1 for Wakeup Context requirements and reset stimuli).

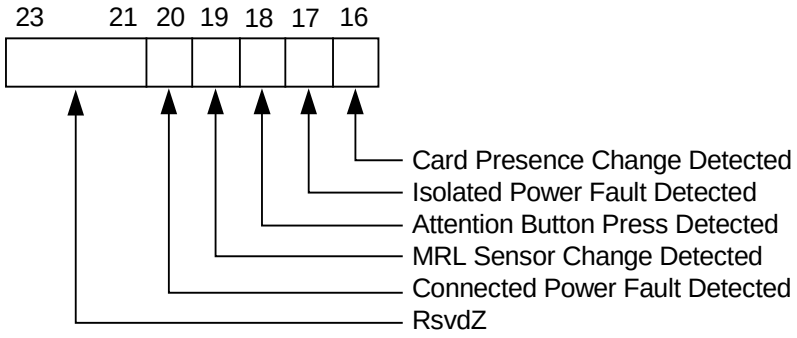



**Figure 4-21: Slot Event Latch Field**

**Table 4-19: Slot Event Latch Field**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 16 | **Card Presence Change Detected** – This bit is set when a change is detected on the **PRSNT1#/PRSNT2#** bits defined in the Slot Status field.<br><br>This bit's default value is 0. | R/WC |
| 17 | **Isolated Power Fault Detected** – This bit is set when an isolated power fault is detected by the power control circuitry for this slot. See Section 3.4.2 for a description of isolated power faults.<br><br>This bit's default value is 0. | R/WC |
| 18 | **Attention Button Press Detected** – This bit is set when the Attention Button bit in the Slot Status field transitions from 0 to 1 indicating the Attention Button has been pressed.<br><br>This bit's default value is 0. | R/WC |
| 19 | **MRL Sensor Change Detected** – This bit is set when the MRL Sensor bit in the Slot Status field changes state indicating a change in the position of the MRL.<br><br>This bit's default value is 0. | R/WC |
| 20 | **Connected Power Fault Detected** – This bit is set when a connected power fault is detected by the power control circuitry for this slot. See Sections 3.4.2 for a description of connected power faults.<br><br>This bit's default value is 0. | R/WC |

### 4.5.12.3. Slot SERR-INT Mask Field

The Slot SERR-INT Mask field controls masking and unmasking of System Interrupts and system errors generated from events detected by the SHPC. See Section 4.7 for a description of using this register to generate System Interrupts and system errors.

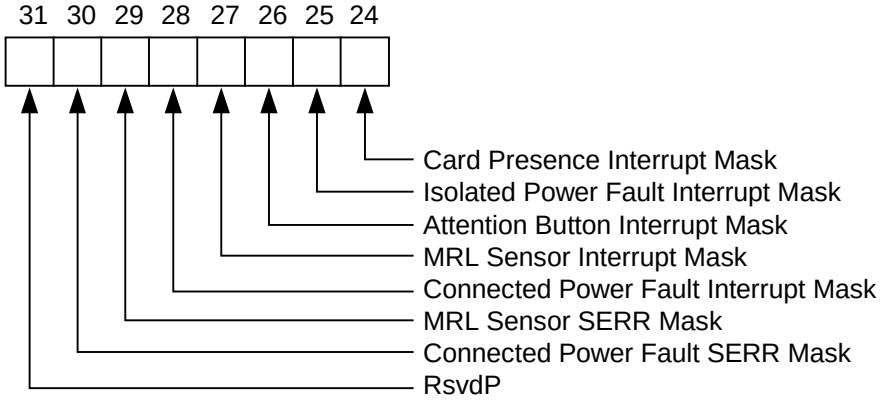



**Figure 4-22: Slot SERR-INT Mask Field**

**Table 4-20: Slot SERR-INT Mask Field**

| Bit Location | Description | Register Attribute |
|---|---|---|
| 24 | **Card Presence Interrupt Mask** – If this bit is set, System Interrupts from Card Presence Change Detected are masked. The state of this bit has no effect on the state of the Card Presence Change Detected bit. (See Section 4.7 for details.)<br><br>This bit has a default value of 1. | R/W |
| 25 | **Isolated Power Fault Interrupt Mask** – If this bit is set, System Interrupts from Isolated Power Fault Detected are masked. The state of this bit has no effect on the state of the Isolated Power Fault Detected bit. See Section 4.7 for details of System Interrupts. See Section 3.4.2 for a description of isolated power faults.<br><br>This bit has a default value of 1. | R/W |
| 26 | **Attention Button Interrupt Mask** – If this bit is set, System Interrupts from Attention Button Press Detected are masked. The state of this bit has no effect on the state of the Attention Button Press Detected bit.<br><br>This bit has a default value of 1. | R/W |
| 27 | **MRL Sensor Interrupt Mask** – If this bit is set, System Interrupts from MRL Sensor Change Detected are masked. The state of this bit has no effect on the state of the MRL Sensor Change Detected bit.<br><br>This bit has a default value of 1. | R/W |

| | | |
|---|---|---|
| 28 | **Connected Power Fault Interrupt Mask** – If this bit is set, System Interrupts from Connected Power Fault Detected are masked. See Section 3.4.2 for a description of connected power faults. The state of this bit has no effect on the state of the Connected Power Fault Detected bit.<br><br>This bit has a default value of 1. | R/W |
| 29 | **MRL Sensor SERR Mask** – If this bit is set, SERR assertions from MRL Sensor Change Detected are masked. The state of this bit has no effect on the state of the MRL Sensor Change Detected bit.<br><br>This bit has a default value of 1. | R/W |
| 30 | **Connected Power Fault SERR Mask** – If this bit is set, SERR assertions from Connected Power Fault Detected are masked. See Section 3.4.2 for a description of connected power faults. The state of this bit has no effect on the state of the Connected Power Fault Detected bit.<br><br>This bit has a default value of 1. | R/W |

## 4.6. Controller Command Support

Software issues a command to the SHPC by writing the appropriate command code to the Command Code field of the Controller Command register. Writing the command to the Command Code field triggers the SHPC to begin executing the command. Table 4-21 shows the bit encoding for each command. All other command codes are reserved. When the SHPC begins executing a command, it must set the Controller Busy bit in the Controller Command Status register to indicate it is busy executing a command. The SHPC must change the busy bit from 0 to 1 immediately when software writes to the Controller Command register. This prevents polling software from reading a 0 before the command begins execution and mistakenly assuming the command has completed. When the SHPC has finished executing the command, it changes the Controller Busy bit from 1 to 0. If the command fails, the SHPC must set the appropriate bit or bits in the Controller Command Error Code field. The behavior for all commands defined by this specification (except vendor specific) requires the SHPC to set only 1 bit in the Controller Command Error Code field if it fails. The SHPC is permitted to set multiple bits in the Controller Command Error Code field for vendor specific commands. If the command is successful, no bits in the Controller Command Error Code field are set. The following sections describe each command and the error cases for each command along with the bit in the Controller Command Error Code field that is set.

The SHPC must set the Invalid Command bit of the Controller Command Status register if any command is issued using a reserved command code.

The SHPC must be able to detect power faults and arbiter timeouts while it is executing a command (see Sections 3.4.2 and 3.4.1). The SHPC must prevent both conditions from affecting the executing command (that is, the command must not fail due to power faults or arbiter timeouts). Software services the power fault or arbiter timeout independent of command execution and completion.

The SHPC must complete commands within the following time limits:

- Slot Operation and Set Bus Segment Speed/Mode commands are required to execute in less than 1 second.
- The Enable All Slots and Power-Only All Slots commands are required to execute in less than 15 seconds.

**Implementation Note: Command Execution Time May Affect System Boot Time**

The time limits shown above are maximum allowable values. However, SHPC designers are strongly urged to provide faster command execution times.

The Enable All Slots and Power-Only All Slots commands can have a significant, detrimental impact on length of system boot time if allowed to extend to the maximum value.

The SHPC may optionally be configured to generate a System Interrupt when a command completes. (See Section 4.7 for further details.)

**Table 4-21: Controller Command Codes**

| | | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|
| **Command** | **Command Code (hex)** | **Command Code Bits** | | | | | | | |
| Slot Operation | 00h – 3Fh | 0 | 0 | Attention Indicator | | Power Indicator | | Slot State | |
| Set Bus Segment Speed/Mode | 40h – 47h | 0 | 1 | 0 | 0 | 0 | Bus Speed/Mode | | |
| Power-Only All Slots | 48h | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| Enable All Slots | 49h | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| Reserved Command Codes | 4Ah - BFh | Reserved Command Codes | | | | | | | |
| Vendor Specific Commands | C0h - FFh | 1 | 1 | Vendor Specific Command | | | | | |

The following sections describe each command listed in Table 4-21.

## 4.6.1. Slot Operation Command

The Slot Operation command changes attributes associated with a slot. Slot attributes include Slot State, Power Indicator State, and Attention Indicator State. The Target Slot field in the Controller Command register selects the target slot for Slot Operation commands (see Section 4.5.7). The format of the Slot Operation command code gives software the ability to change one, two, or all three slot attributes by issuing one command. The format of the Slot Operation command is shown in Table 4-22. The lower six bits of the Slot Operation command code contain three, two-bit fields. Each two-bit field corresponds to an attribute.

**Table 4-22: Format of Slot Operation Command**

| 7 | | | | | | | 0 | **Slot State** |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | X | X | X | X | 0 | 0 | No Change |
| 0 | 0 | X | X | X | X | 0 | 1 | Power Only |
| 0 | 0 | X | X | X | X | 1 | 0 | Enable |
| 0 | 0 | X | X | X | X | 1 | 1 | Disable |
| | | | | | | | | **Power Indicator State** |
| 0 | 0 | X | X | 0 | 0 | X | X | No Change |
| 0 | 0 | X | X | 0 | 1 | X | X | On |
| 0 | 0 | X | X | 1 | 0 | X | X | Blink |
| 0 | 0 | X | X | 1 | 1 | X | X | Off |
| | | | | | | | | **Attention Indicator State** |
| 0 | 0 | 0 | 0 | X | X | X | X | No Change |
| 0 | 0 | 0 | 1 | X | X | X | X | On |
| 0 | 0 | 1 | 0 | X | X | X | X | Blink |
| 0 | 0 | 1 | 1 | X | X | X | X | Off |

Note: X = Don't Care

When a Slot Operation command is issued, the value in each two-bit field determines the new state that the attribute transitions to when the command completes successfully. If a particular attribute's two-bit field is set to 00b, the state of the attribute must not change when the command is executed by the SHPC. Additionally, if the two-bit field associated with an attribute is set to a value that matches the state the attribute is currently in, the attribute state must not change. In both of these cases, the SHPC performs no operation for that particular field. This gives software the ability to change the state of multiple attributes associated with a slot by issuing a single command and not affect existing attribute states. For example, the following command would be issued to the SHPC to enable the slot and turn on the Power Indicator LED but not change the state of the Attention LED Indicator (see Figure 4-23). If the slot is already in the enabled state when the command is issued, no slot enable or disable sequence is executed.

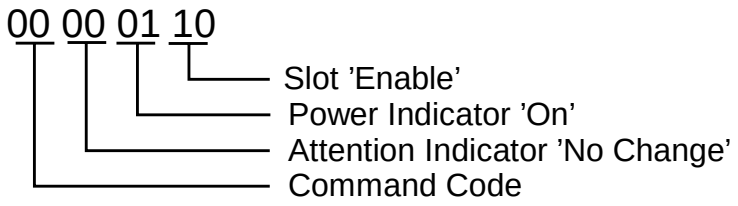



**Figure 4-23: Example Slot Operation Command**

If all two-bit fields in the Slot Operation command are either 00b or the same as the current attribute state, the command performs no operation. The SHPC successfully completes the command, none of the attributes is affected, and the slot is not affected in any way.

A Slot Operation command issued when Target Slot is set to 0 fails and sets the Invalid Command bit in the Controller Command Error field. Also, if Target Slot is greater than the Number of Slots Implemented field in the Slot Configuration register, the command must fail and set the Invalid Command bit.

If any Slot Operation command fails, the states of the slot's attributes must be left unchanged.

### Slot State Commands

The following commands are issued to change the state of a slot:

- Power Only – This command powers up the target slot but does not connect the clock or bus signals (see Section 3.4.5). Software uses this command to determine if a card is capable of running at 66 MHz conventional mode before enabling it (see Section 4.5.12.1).

  If the Target Slot field in the Controller Command register is greater than the Number of Slots Implemented field in the Slot Configuration register, the command must fail and the Invalid Command bit in the Controller Command Error Code field must be set.

  The slot must be disabled (or powered-only) when software issues this command. If the slot is currently enabled and this command is issued, the command must fail and the Invalid Command bit of the Controller Command Error Code field must be set.

  If the MRL of the target slot is open, the command must fail and the MRL Open bit of the Controller Command Error Code field must be set.

- Enable – This command powers up the target slot, enables the clock, and connects the bus signals (see Section 3.4.3). At this point, the card is fully functional from a hardware standpoint. Software must still configure the card and load a driver for it.

  If the card is not capable of running at the speed or mode that the bus is currently running, the command must fail and the Invalid Speed/Mode bit in the Controller Command Error code field must be set.

  If the Target Slot field in the Controller Command register is greater than the Number of Slots Available field (for the current bus speed and mode) in the Slots Available registers, the command must fail and the Invalid Command bit in the Controller Command Error Code field must be set. See Section 4.5.2 for details on how the SHPC handles bus-loading restrictions. The Enable command is permitted to target any slot that is less than or equal to the Number of Slots Available, regardless of the present state of that slot.

  If the MRL of the target slot is open, the command must fail and the MRL Open bit of the Controller Command Error Code field must be set.

- Disable – This command disables the slot as described in Section 3.4.4. The card is safe to remove from the system after this command has completed. The Disable command is permitted to target any slot that is less than or equal to the Number of Slots Implemented field in the Slot Configuration register, regardless of the present state of that slot.

  If the Target Slot field in the Controller Command register is greater than the Number of Slots Implemented field in the Slot Configuration register, the command must fail and the Invalid Command bit in the Controller Command Error Code field must be set.

### Power and Attention Indicator State Commands

The following commands are issued to change the state of an indicator:

- On – This command turns the indicator on solid.
- Blink – This command causes the indicator to blink.
- Off – This command causes the indicator to turn off.

The indicators are always under the control of software and their state is changed only as a result of issuing a command to the SHPC. For example, if the user opens the MRL and the slot is automatically powered down by the SHPC, the indicators are unaffected. In this case, software must issue a Slot Operation command to turn off the Power Indicator.

See Section 2.2.1 for a discussion on how indicators are used in the Standard Usage Model and for specifications on indicator blink rate and duty cycle.

### 4.6.2. Set Bus Segment Speed/Mode

The Set Bus Segment Speed/Mode command changes the mode and/or speed of the bus that contains the slots controlled by the SHPC. Issuing this command causes the bus segment to be reset and the requested bus speed and mode to take affect. Because add-in cards in the bus and its subordinate buses are reset by this command, software must quiesce all affected devices before issuing this command.

The lower three bits of the Command Code register determine what new speed and mode the bus transitions to when the command completes (see Table 4-23).

**Table 4-23: Format for Set Bus Segment Speed/Mode Command**

| 2 | 1 | 0 | Requested Speed and Mode |
|---|---|---|---|
| 0 | 0 | 0 | 33 MHz, Conventional Mode |
| 0 | 0 | 1 | 66 MHz, Conventional Mode |
| 0 | 1 | 0 | 66 MHz, PCI-X Mode |
| 0 | 1 | 1 | 100 MHz, PCI-X Mode |
| 1 | 0 | 0 | 133 MHz, PCI-X Mode |
| 1 | 0 | 1 | Reserved |
| 1 | 1 | 0 | Reserved |
| 1 | 1 | 1 | Reserved |

Software must make sure all add-in cards on the bus segment are capable of running at the requested speed before using this command. Card capabilities are determined using the 66 MHz Capable and the PCI-X Capable fields in the Slot Status field (see Section 4.5.12).

If any enabled add-in card on the bus segment is not capable of running at the requested speed or mode, this command must fail, and the Invalid Speed/Mode bit in the Controller Command Error Code field must be set.

If this command is issued to transition the bus segment to a speed/mode that would require a currently enabled add-in card to be disabled due to bus loading restrictions (as indicated by the appropriate Number of Slots Available field), the command must fail and the Invalid Speed/Mode bit must be set (see Section 4.5.2). For example, if all of the following are true:

- the bus segment is currently running at 100 MHz PCI-X mode
- the bus segment contains two enabled cards
- the Number of Slots Available (133 MHz PCI-X) field contains 1

then a command to change the bus to 133 MHz PCI-X mode must fail because the bus is not capable of supporting both enabled cards at 133 MHz PCI-X mode.

If the Set Bus Segment Speed/Mode command is issued for a segment and the Number of Slots Available register described in Section 4.5.2 indicates no slots are available at the

requested speed/mode, the command must fail and the Invalid Command bit in the Controller Command Error Code field of the Controller Command Status register must be set. For example, if a bus segment is capable of 100 MHz PCI-X, the Number of Slots Available (100 MHz PCI-X) field would contain the number of slots that are permitted to be enabled on this bus segment. If this field is zero, the segment is not capable of running at 100 MHz PCI-X and a command to switch to it would fail as described above.

If this command fails, the speed and mode of the bus must not change and cards must not be reset or affected in any way.

Issuing this command with a value that matches the current speed/mode of the bus segment has the effect of resetting the bus. The speed and mode of the bus is not affected but add-in cards must be reset.

All slot attributes must stay the same on completion of this command regardless of whether the command fails or succeeds.

The Target Slot bits in the Controller Command register must be ignored for Set Bus Segment Speed/Mode commands.

### 4.6.3. Power-Only All Slots

The Power-Only All Slots command changes the state of all slots controlled by the SHPC (that have their MRL closed) to “powered only” and changes the state of their Power Indicators to on. The number of slots controlled by the SHPC is determined by the Number of Slots Implemented field in the Slot Configuration register. Slots with their MRL open remain in the disabled state and their Power Indicator state is set to off (regardless of the indicators previous state). If all slots controlled by the SHPC have their MRLs open, the Slot States must remain disabled and the Power Indicator states must be set to off. (The command must complete successfully even if all the MRLs associated with controlled slots are open).

If one or more slots controlled by the SHPC are enabled when this command is issued, the command must fail. No slot states or LED states are changed, and the Invalid Command bit contained in the Controller Command Error Code field is set.

### 4.6.4. Enable All Slots

The Enable All Slots command changes the state of all slots available at the current bus speed/mode (that have their MRL closed) to enabled and changes the state of their Power Indicator to on. See Section 4.5.2 for a description of all slots available at the current bus speed/mode. Slots with their MRL open remain in the disabled state and their Power Indicator state is set to off (regardless of the indicators previous state). If all slots available at the current bus speed/mode have their MRLs open, the slot states must remain disabled and the Power Indicator states must be set to off. (The command must complete successfully, even if all the MRLs associated with controlled slots are open.)

If one or more slots available at the current bus speed/mode are enabled when this command is issued, the command must fail. No slot states or LED states are changed, and the Invalid Command bit contained in the Controller Command Error Code field is set.

If one or more cards in an available slot at the current bus speed/mode has its MRL closed and does not support the present mode/speed of the bus, the command must fail and set the Invalid Speed/Mode bit in the Controller Command Error Code field.

This command can result in an "Invalid mode/speed" error in two cases:

- The user closes an MRL on a slot after software has checked mode/speed but before the Enable All Slots command is issued and that slot is not capable of running at the requested speed/mode.
- Software changes the bus mode/speed after the add-in card mode/speeds are checked, and an add-in card in a disabled slot does not support the new bus speed/mode.

### 4.6.5. Vendor Specific Commands

Vendor Specific commands are specific to a particular implementation of the SHPC. As such, the details of these commands and their failure conditions are not defined in this specification. Vendor Specific commands may optionally set any number of bits in the Controller Command Error Code field (see Section 4.5.8). Vendor Specific command codes must be confined to C0h-FFh (see Table 4-21).

Some examples of possible uses of Vendor Specific commands include following:

- Device and System Diagnostics
- Custom Modes and Features

There must not be a requirement for software to issue a Vendor Specific command for proper SHPC behavior.

## 4.7. SHPC Event Processing

The SHPC detects the following events:

- Slot Events (on the slots it controls):
  - Attention Button Depression
  - Isolated Power Fault (see Section 3.4.2)
  - Connected Power Fault (see Section 3.4.2)
  - Card Presence Change
  - MRL Sensor Change
- Controller Events
  - Command Completion (see Section 4.6)
  - Arbiter Timeout (see Section 3.4.1)

### 4.7.1. Slot Events

The SHPC monitors all the slots it controls for the slot events listed above. When the SHPC detects one of the slot events listed above, it sets the latch bit in the Slot Event Latch field that corresponds to the event (see Section 4.5.12.2). At that point, the event is pending until software clears the event by writing a 1 to the Slot Event Latch field bit that corresponds to the event.

Once a slot event is pending on a particular slot, all subsequent events of that type are ignored on that slot until the event is cleared. The SHPC must continue to monitor the slot for all other slot event types. For example, if the SHPC detects that the Attention Button is pressed, the Attention Button Press Detected bit in the Slot Event Latch field is set. The SHPC ignores all subsequent Attention Button presses on that slot until the

event is cleared. The SHPC continues to monitor all other types of slot events, and if one is detected, the latch bit associated with the newly detected event is set as well (that is, multiple bits in the Slot Event Latch field are set if multiple slot events are detected).

Upon detection of a power fault, software must issue a Slot Disable command (see Section 4.6.1) targeted to the slot reporting the fault. Power faults (both isolated and connected) are latched not only in the Slot Event Latch register but also in the Power Controller. The Power Controller clears its main power fault latch when power to the slot is turned off, and clears its auxiliary power fault latch when the slot's MRL is opened as described in Section 3.1.2.2. Software determines whether a fault is on main power or auxiliary power by observing the state of the Power Fault bit in the Slot Status field when the slot is in the disabled state. An auxiliary power fault is indicated if the slot is in the disabled state and the Power Fault bit in the Slot Status field is set. When an auxiliary power fault has been identified, it is recommended that software notify the user that an auxiliary power fault exists on the slot and recommend opening the MRL to clear the fault.

### 4.7.2. Controller Events

When the SHPC completes execution of a command, the Command Completion Detected bit in the Controller SERR-INT register is set (see Section 4.5.11). Software clears the event by writing a 1 to the Command Completion Detected bit. The SHPC must continue to process commands normally when a Command Completion Detected event is pending. For example, if software issues a command to the controller and it completes, the Command Completion Detected bit is set. If software then issues another command without clearing the previous Command Completion Detected event, the SHPC must process the command normally (see Section 4.6 for a description of how the SHPC processes commands).

When the SHPC detects an arbiter timeout (see Section 3.4.1), the Arbiter Timeout Detected bit in the Controller SERR-INT register is set. Software writes a 1 to the Arbiter Timeout Detected bit to clear the pending event.

### 4.7.3. Event Polling and Notification

Depending on how the SHPC is configured, software polls for or is notified of pending event conditions. Each event listed above may optionally be programmed to generate a System Interrupt, assert the Wakeup Signal, or pulse **SERR#** (see Table 4-24).

**Table 4-24: Event Behavior**

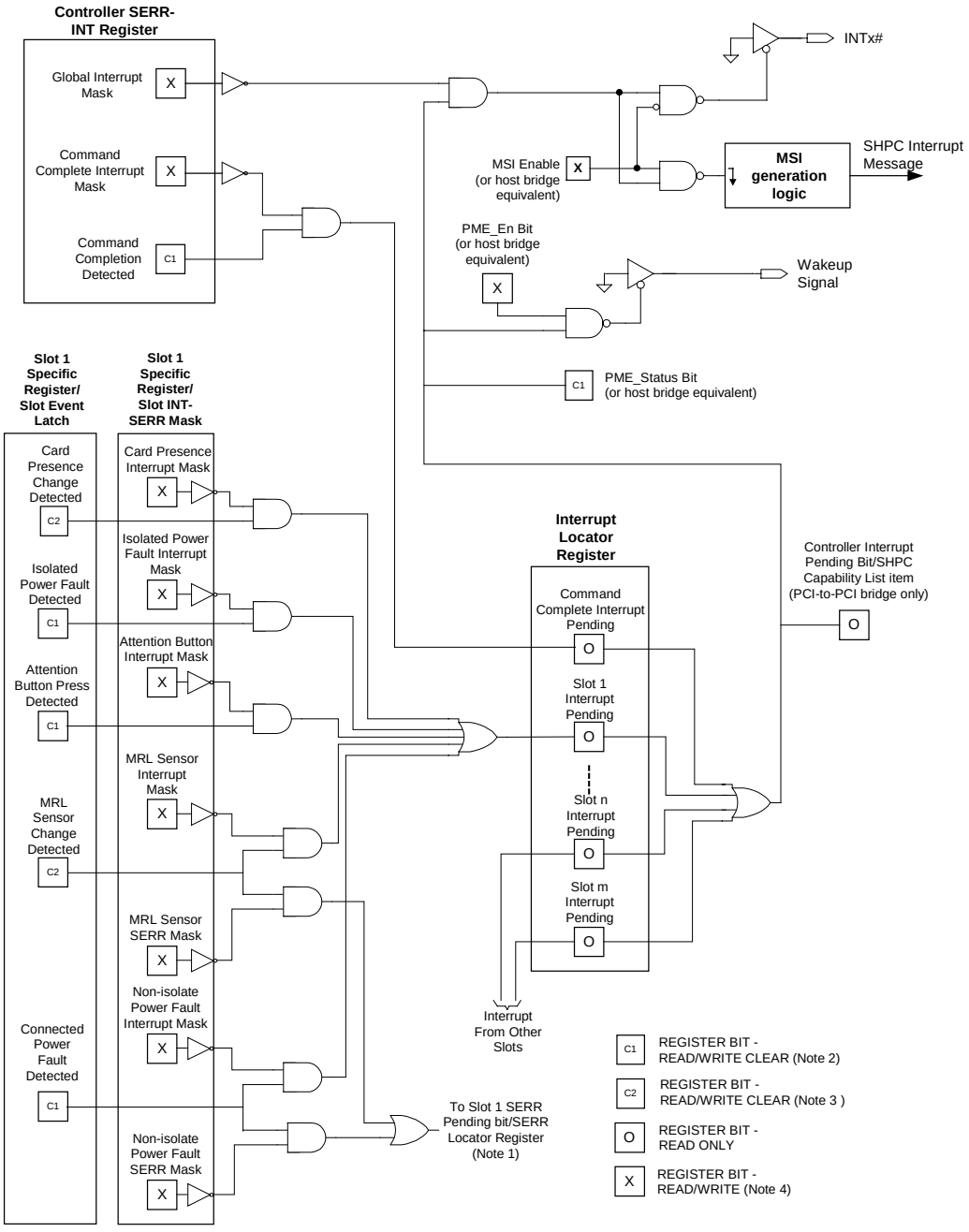
| Event | Register Bit Set When Detected | Cleared by | SHPC Optionally Generates the Following When Event is Detected |
|---|---|---|---|
| **Slot Events** | | | |
| Attention Button Press | Attention Button Press Detected bit / Slot Event Latch field | Writing a 1 to the detected bit | Wakeup Signal, System Interrupt |
| Isolated Power Fault | Isolated Power Fault Detected bit / Slot Event Latch field | Writing a 1 to the detected bit. Issue 'Slot Disable' command to slot. (Note). | Wakeup Signal, System Interrupt |
| Card Presence Change | Card Presence Change Detected bit / Slot Event Latch field | Writing a 1 to the detected bit | Wakeup Signal, System Interrupt |
| MRL Sensor Change | MRL Sensor Change Detected bit / Slot Event Latch field | Writing a 1 to the detected bit | Wakeup Signal, System Interrupt, **SERR#** |
| Connected Power Fault | Connected Power Fault / Slot Event Latch field | Writing a 1 to the detected bit. Issue 'Slot Disable' command to slot. (Note) | Wakeup Signal, System Interrupt, **SERR#** |
| **Controller Events** | | | |
| Command Completion | Command Completion Detected bit / Controller SERR-INT register | Writing a 1 to the detected bit | Wakeup Signal, System Interrupt |
| Arbiter Timeout | Arbiter Timeout Detected bit / Controller SERR-INT register | Writing a 1 to the detected bit | **SERR#** |

Note: If the source of the power fault indication is a fault on **3.3Vaux**, the Power Fault bit in the Slot Status register is cleared only by opening the MRL. (See Section 3.1.2.2.)

The following sections describe System Interrupt and system error generation and processing.

### 4.7.3.1. System Interrupts

Throughout this section refer to Figure 4-24, which shows the logical connection between event processing register bits related to the System Interrupt and the Wakeup Signal in the SHPC Working Register set.




Notes:

1. See Figure 4-25.
2. See Figure 4-26.
3. See Figure 4-27.
4. See Figure 4-28.

**Figure 4-24: System Interrupt/Wakeup Signal Logic**

A bridge integrated with an SHPC optionally supports the Message Signaled Interrupts capability defined in PCI 2.2, Section 6.8, for PCI-to-PCI bridges or an equivalent host-

bridge mechanism. If a bridge supports Message Signaled Interrupts, and the MSI Enable bit in the Message Control register in the Message Capability Structure (see PCI 2.2, Section 6.8.1.3 or equivalent host bridge mechanism) is set, the bridge generates a System Interrupt by sending an interrupt message. If a bridge does not support Message Signaled Interrupts, or the MSI Enable bit is not set, the bridge generates a System Interrupt by asserting its **INTx#** pin. The bridge is recommended to support an **INTx#** pin, even if the Message Signaled Interrupt capability is supported, to provide compatibility with systems that do not support Message Signaled Interrupts.

### Implementation Note: Message Signaled Interrupts and Host Bridges

Message Signaled Interrupts as defined in PCI 2.2 are controlled by a data structure in the Capabilities List, and thus apply only to devices that appear in Configuration Space, for example, PCI-to-PCI bridges. This specification defines host bridge registers only in the Host Bridge Register Block in Memory Space (see Section 4.3.2.1), and does not have a defined method for controlling Message Signaled Interrupts. Some host bridges also appear in Configuration Space. However, since PCI 2.2 does not specify the characteristics of host bridge Configuration Space registers, these registers cannot be used to control the host bridge Message Signaled Interrupt capability either (as is also true of the power management capability).

Although no Message Signaled Interrupt capability is presently defined in the Host Bridge Register Block, this specification allows an equivalent host bridge mechanism for Message Signaled Interrupts, in anticipation that the capability will be added to the Host Bridge Register Block at a future date.

An interrupt is said to be “active” if an event is detected and all its associated mask bits are cleared. If the SHPC is designed and enabled to generate a System Interrupt by asserting **INTx#**, the SHPC asserts **INTx#** whenever at least one interrupt is active. In other words, the SHPC asserts **INTx#** if Global Interrupt Mask is cleared and one or more bits in the Interrupt Locator registers are a 1. Software clears the interrupt by clearing (or masking) all pending slot events on all slots controlled by the SHPC and Command Completion Detected. This clears all the bits in the Interrupt Locator register and the Controller Interrupt Pending bit (in a PCI-to-PCI bridge) and causes the SHPC to stop asserting **INTx#**. The **INTx#** pin is, by nature a level-triggered interrupt.

If the SHPC is designed and enabled to generate a System Interrupt by signaling a Message Signaled Interrupt, the SHPC signals an interrupt message each time its interrupt state changes from the “no interrupts active” state to a state in which one or more interrupts are active. The SHPC does not signal an interrupt message if one or more interrupts become active while another interrupt is already active. All active interrupts must be cleared or masked by the software before the SHPC will send another interrupt message. Message Signaled Interrupts are, by nature, edge-triggered events.

If a bridge integrated with an SHPC supports Message Signaled Interrupts, the SHPC Interrupt Message Number register indicates to the software which interrupt message is used by the SHPC. If the bridge includes sources of interrupts other than the SHPC, the bridge must allow system software to assign the SHPC a unique interrupt message. In this case, the bridge must request multiple messages, as defined in PCI 2.2. If system software allocates fewer interrupt messages to such a bridge than it requests, system software must resolve issues dealing with sharing a single edge-triggered interrupt between independent interrupt service routines. These issues are beyond the scope of this specification.

### Implementation Note: Servicing Message Signaled Interrupts

As shown in Figure 4-24, the hardware design of the Message Signaled Interrupt generation logic signals an interrupt message any time a falling edge would have occurred on the **INTx#** pin, if Message Signaled Interrupts had not been enabled. For example, consider a scenario in which two interrupt events occur in rapid succession. The first event causes the Message Signaled Interrupt to be signaled. Before the interrupt service routine has had an opportunity to service and clear (or mask) the first event, the second event occurs. In this case, only one Message Signaled Interrupt is signaled, because the first event is still active at the time the second event occurs (a hardware **INTx#** pin signal would have had only one falling edge). For another example, consider the case in which an interrupt is active and software sets the Global Interrupt Mask bit. If software clears the Global Interrupt Mask bit without clearing the source of the interrupt, the SHPC signals a Message Signaled Interrupt again (a hardware **INTx#** pin signal would have had another falling edge).

As described in PCI 2.2, the SHPC and the system software must implement a handshake to guarantee that all interrupt events are serviced. One alternative implementation of such a handshake is for the interrupt service routine to set the Global Interrupt Mask bit before it clears any interrupting events and to clear the Global Interrupt Mask bit when it has cleared all the interrupting events it intends to service. (This could be any number of events from one to the total number of pending events.) If all the events are cleared, clearing the Global Interrupt Mask bit prepares the SHPC to signal a Message Signaled Interrupt the next time it detects an interrupting event. If one or more events are still pending at the time the Global Interrupt Mask bit is cleared, the SHPC immediately signals another Message Signaled Interrupt.

Another alternative handshake implementation to guarantee that all interrupt events are serviced is for the interrupt service routine to re-inspect the SHPC Interrupt Locator register after clearing or masking what is presumed to be the last pending interrupt. If another event is found to be active, it is serviced in the same interrupt service routine. This insures that, if an additional interrupting event occurs before the previous interrupt is cleared (so that the SHPC does *not* send an additional interrupt message), that new event is serviced as part of the previous interrupt service routine.

This alternative has the side effect, in some cases, of servicing an additional event in the interrupt service routine that has already signaled a new Message Signaled Interrupt. In some such cases, when the new interrupt is serviced, no interrupts are active. Such conditions are sometimes referred to as spurious interrupts. To illustrate, assume that software has completed its interrogation of all interrupt sources from the bridge and has actually cleared the last active interrupt. Further assume that, before software re-checks the Interrupt Locator register to guarantee that the last interrupt has been cleared, another (new) interrupt event occurs. Because this new event occurred *after* all previous interrupts were cleared, the SHPC signals a new interrupt message. However, since the event occurred *before* the service routine for the initial interrupt had finished, the first invocation of the service routine discovers and services the new interrupt event as well. In this case, when the software responds to the second Message Signaled Interrupt, it will not find any active interrupts.

See Section 6.3 for a description of Wakeup Signal generation.

The Controller Interrupt Pending bit located in the SHPC Capabilities List Item for a PCI-to-PCI bridge is a 1 if one or more bits in the Interrupt Locator register are a 1. This gives software the ability to quickly determine in Configuration Space if the SHPC has detected any events (independent of the Global Interrupt Mask). This bit is 0 if all bits in the Interrupt Locator register are 0. The same information is determined in Memory

Space for a PCI-to-PCI bridge and for a host bridge by reading the Interrupt Locator register directly.

Each slot controlled by the SHPC has an associated bit in the Slot n Interrupt Pending field of the Interrupt Locator register (see Section 4.5.9 for more details on the Interrupt Locator register). When read, these bits return 1 if one or more of the slot events listed above are detected by the SHPC and the corresponding interrupt mask bits in that slot's Slot SERR-INT Mask field are cleared. For example, if the SHPC detects an Attention Button press on slot 2, the following would occur:

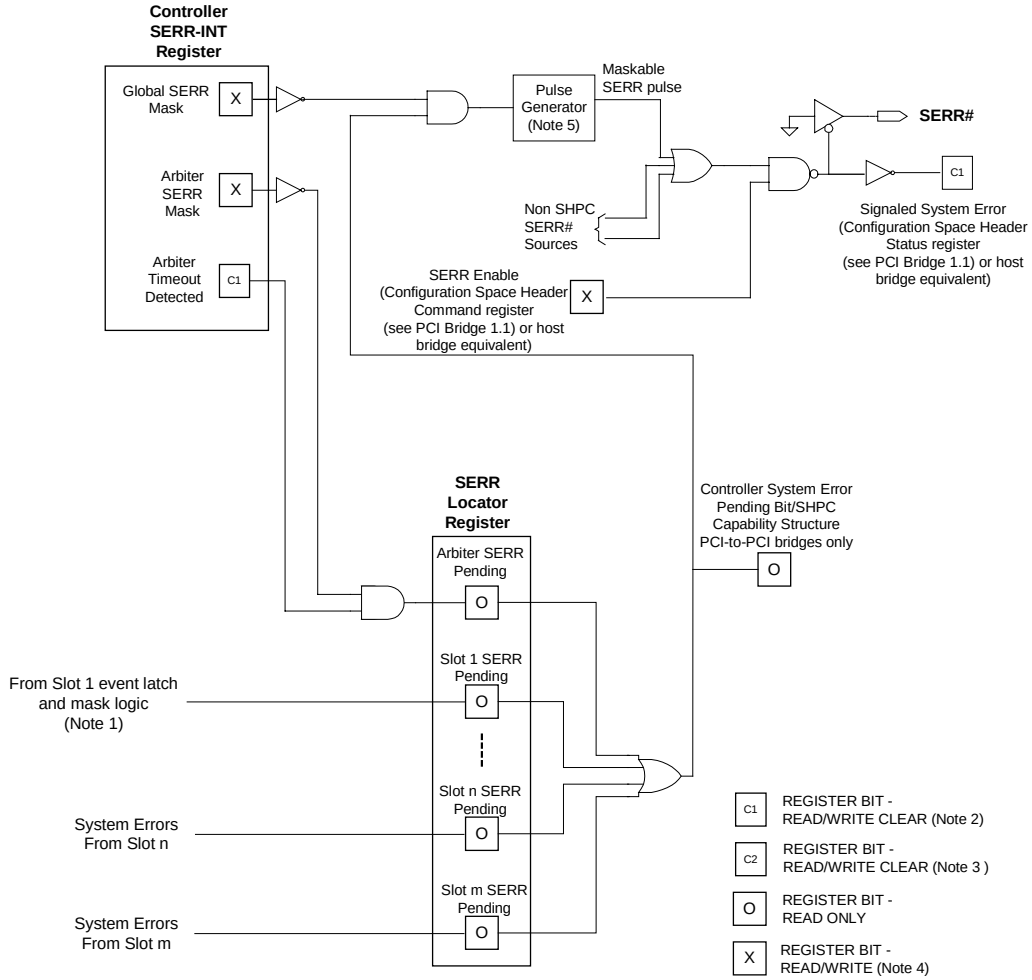
- Slot 2's Attention Button Press Detected bit, located in the Slot Event Latch field, is set.
- If slot 2's Attention Button Interrupt Mask bit is cleared, bit 2 of the Interrupt Locator register is a 1, indicating that slot 2 has an interrupt condition pending.
- If the Global Interrupt Mask is cleared, the SHPC generates a System Interrupt (as described above).

To clear a slot's bit in the Slot n Interrupt Pending field, software clears or masks all pending slot events for that slot. Clearing an event is accomplished by writing a 1 to the appropriate bit in the Slot Event Latch field and masking an event is accomplished by writing a 1 to the appropriate bit in the Slot SERR-INT Mask field.

When the Command Complete Interrupt Pending bit is read, it returns 1 if the Command Complete Detected bit is set and the Command Complete Interrupt Mask bit is cleared. Clearing a pending Command Completion Detected event (by writing a 1 to the Command Completion Detected bit) or masking the condition (by setting the Command Complete Interrupt Mask bit) clears the Command Complete Interrupt Pending bit.

### 4.7.3.2. System Errors

Throughout this section, refer to Figure 4-25, which shows the logical connection between event processing register bits related to **SERR#** in the SHPC Working Register set.




Notes:

1. See Figure 4-24.
2. See Figure 4-26.
3. See Figure 4-27.
4. See Figure 4-28.
5. See Figure 4-29.

**Figure 4-25: SERR# Logic**

Each slot controlled by the SHPC has an associated bit in the Slot n SERR Pending field of the SERR Locator Register (see Section 4.5.10 for more details on the SERR Locator Register). When read, these bits return 1 if a connected power fault or MRL Sensor change is detected and the corresponding mask bit in the Slot SERR-INT Mask field is cleared. For example, if the SHPC detects a connected power fault on slot 2, the following would occur:

- Slot 2's Connected Power Fault Detected bit (located in the Slot Event Latch field) is set.
- If slot 2's Connected Power Fault SERR Mask bit is cleared, bit 2 of the SERR Locator register is a 1, indicating that slot 2 has an **SERR#** condition pending.
- If the Global SERR Mask is cleared, **SERR#** is pulsed (see details below).

### Implementation Note: Programming SERR# on MRL Opening

Since the action of **SERR#** in a system is usually catastrophic (the system is halted), system software developers need to be careful to enable **SERR#** generation only when it really is protecting against potential data corruption or anomalous system behavior. An example of a case in which asserting **SERR#** would not be appropriate is an unexpected opening of an MRL for an empty slot. Opening the MRL when a slot is empty has no bad effects on a system. The slot merely powers down in a controlled fashion. The MRL Sensor SERR Mask bit in the Slot INT-SERR Mask register should be set when a slot is empty. Opening the MRL is also not a problem when a user has begun powering on a new card but the slot has not yet been enabled. **SERR#** generation on MRL opening should never be enabled until just before the Slot Enable command is issued.

In addition to the above obvious case, an analysis of the usage and states of an add-in card and its driver shows that generating **SERR#** on MRL opening is not necessary under many other conditions. If no bus transactions are in progress to or from a device, no inherent problems can be created from opening the MRL. The thing that should be considered is whether system software is written in such a way so as to prevent anomalous system behavior or reading of bad data when a device is unexpectedly removed. PCI-to-PCI bridges are required to return all ones if a target device does not respond (Master-Abort) and to set the Master-Abort status bit in the bridge. See PCI Bridge 1.1, Section 3.2.5.17. If a device driver is written such that every read from the device's registers is checked for returning FFh, then software can detect when a device is unexpectedly not responding. This applies, of course, only to registers where returning FFh is not a valid response.

The third area of consideration should be applied based on the type of device. If a bus master is performing an operation and is unexpectedly removed, the operation merely does not complete but no immediate data corruption occurs. System software can handle this by timeouts on incomplete operations and then reporting the unexpected removal. If a device is being written to, and is unexpectedly removed, no data corruption occurs within the rest of the system. However, corruption can occur in data that is being written to a storage medium or across a wire. If the system software and associated media have guaranteed delivery mechanisms, this is also not necessarily a condition that should halt the system. If the type of device is one that must be polled or data is read from it, and system software cannot verify the validity using the described FFh method above, it would be appropriate to assert **SERR#** to prevent bad data from being propagated after an MRL opened unexpectedly.

System software developers should determine how these conditions are handled in their environments and enable the assertion of **SERR#** if an MRL opens at times when they have determined that bringing the system down is better than other ways of handling the condition. Assertion of **SERR#** from MRL opening should be masked at other times because halting a system unnecessarily can also cause expensive data loss.

To clear a slot's bit in the Slot n SERR Pending field, software must clear the pending MRL Sensor Change Detected bit and Connected Power Fault bit in the slot's Slot Event Latch field, or set the corresponding mask bits.

When the Arbiter SERR Pending bit is read, it returns 1 if the Arbiter Timeout Detected bit is set and the Arbiter SERR Mask bit is cleared. Clearing a pending Arbiter Timeout Detected event (by writing a 1 to the Arbiter Timeout Detected bit) or masking the condition (by setting the Arbiter SERR Mask bit) clears the Arbiter SERR Pending bit.

The Global SERR Mask bit controls pulsing **SERR#** from the SHPC. The SHPC pulses **SERR#** if the SERR# Enable bit in the bridge/SHPC Command register of a PCI-to-PCI bridge (see PCI Bridge 1.1) or host bridge equivalent is set and either of the following is true:

- The Global SERR Mask bit is 0, and the Controller System Error Pending bit changes from 0 to 1.
- The Controller System Error Pending bit is set, and the Global SERR Mask bit changes from 1 to 0.

For example, if one or more bits are set in the SERR Locator register and the Global SERR Mask is set, the SHPC must not pulse **SERR#**. If software then clears the Global SERR Mask bit (and the SERR Enable bit in the Command register is set), the SHPC must pulse **SERR#**.

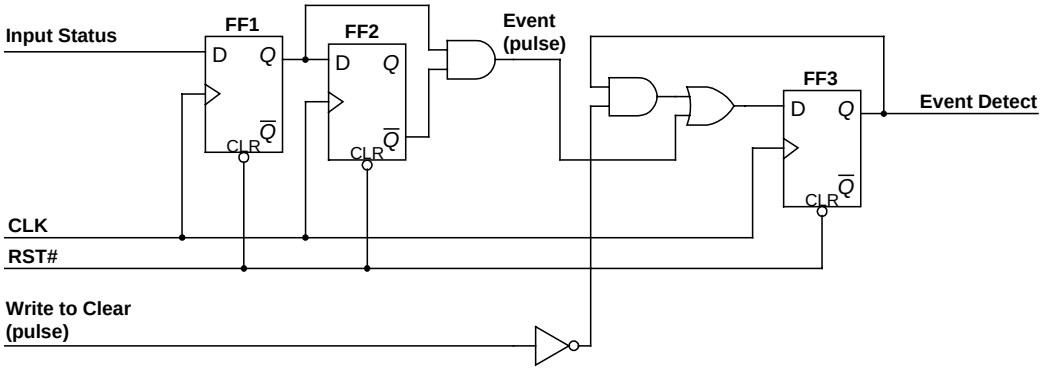



**Figure 4-26: Event Detection Block C1 Circuit**

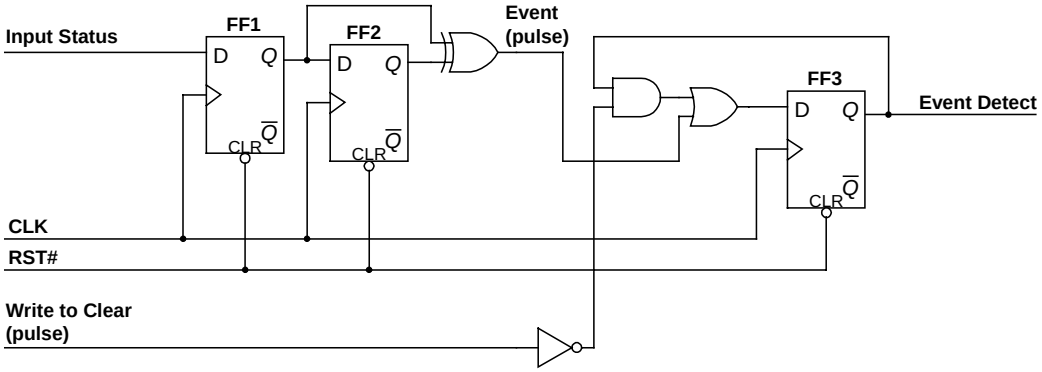



**Figure 4-27: Event Detection Block C2 Circuit**

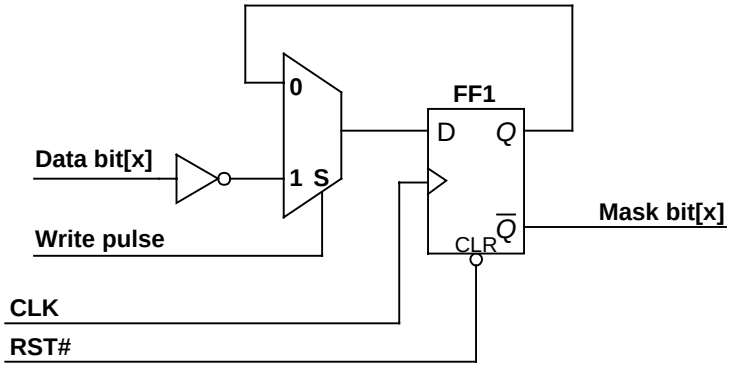



**Figure 4-28: Mask Bit Block X Circuit**

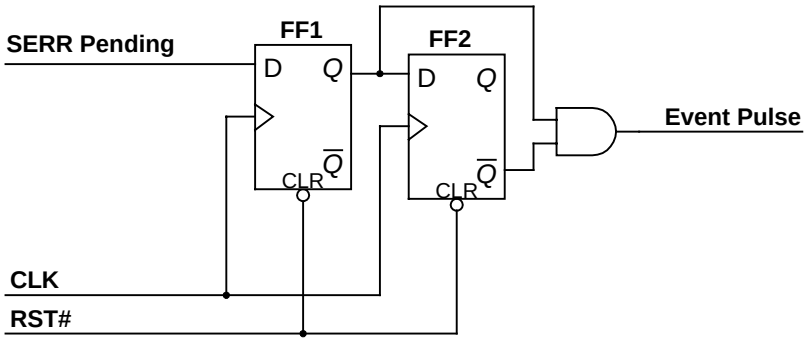



**Figure 4-29: Pulse Generator Block Circuit**

## 4.8. Automatic Power Down

If a slot is in the enabled or "powered only" state when the MRL is opened, the SHPC automatically changes the slot state to disabled. This is done to protect the hardware and operator. If a slot is powered only (not enabled), power is automatically removed by execution of a single-phase operation (see Section 3.4.5). If the slot is enabled when the MRL is opened, the SHPC automatically executes a slot disable sequence (see Section 3.4.4). In either case, no automatic change occurs to the state of either the Power Indicator or Attention Indicator. Software intervention is required to change the states of the indicators to conform to the Standard Usage Model (see Section 2.2.1).

# 5. Firmware

This chapter defines the role of firmware in a hot-plug system. It applies to x86 industry standard architecture and DIG64 1.0 compliant Platforms running operating systems intended for multiple Platform vendors. It additionally assumes that there is at least one SHPC with one or more hot-plug slots (referred to collectively as a hot-plug segment) in the system.

System firmware in a hot-plug system has three roles:

- System initialization. Firmware initializes the hot-plug segments. (See Section 5.1.)
- Firmware services and information. Firmware provides services and information tables that are available when the operating system has been loaded and is executing. (See Section 5.2.)
- Hibernation. On some Platforms, firmware controls how a system enters and resumes from hibernation. (See Section 5.3.)

This specification describes what is required of or recommended for the firmware in each of these three roles.

## 5.1. Initialization of Hot-Plug Segments

A hot-plug system must provide a mechanism for the firmware or software to generate a hardware reset for each PCI bus segment.

Some SHPCs are designed to be initialized by firmware, when a system incorporating hot-plug slots is initially powered up or after a software-initiated system reset. Other SHPCs must be fully initialized by hardware, as specified in Section 3.5.1.3. After the SHPC is initialized, firmware initializes the devices on the hot-plug bus segments. The following steps list the minimum initialization requirements for the firmware in a system with an SHPC, beyond what is normally required of firmware in a non-SHPC system.

1. For PCI-to-PCI bridges (Type 01h Configuration Space header), configure all bridges leading to the hot-plug segment so that the processor can access the Configuration Space of the SHPC. For host bridges with integrated SHPCs, the firmware must initialize the host bridges as necessary to access the Host Bridge Register Block and the SHPC.
2. Firmware initializes the SHPC's hardware-initialized registers, secondary bus frequency and mode, and slot states in the cases specified in Section 3.5.1.3. That section also describes the steps necessary to perform this operation. In all other cases, the bridge integrated with the SHPC initializes the hardware-initialized registers, secondary bus frequency and mode, and slot states completely with hardware.
3. Allocate system resources to the add-in cards and update the HPRT and/or ACPI tables. (See Sections 5.2.2, 5.4, and 5.5.)

During system initialization, hot-insertion and hot-removal operations are not recommended. The results of such operations are unpredictable.

## 5.2. Firmware Services and Information

Firmware Services and Information consist of the following:

- The standard PCI BIOS services as described in the PCI BIOS 2.1 and Section 5.2.1.
- A mechanism to describe the allocation of system resources. (See Section 5.2.2.)
- A mechanism to describe host bridges to facilitate their location by an operating system. (See Section 5.2.3.)

### 5.2.1. PCI BIOS

Firmware on x86 hot-plug Platforms provides BIOS services as described in PCI BIOS 2.1 with the following clarifications:

- The Get PCI Interrupt Routing Options function applies to all hot-plug slots whether they are occupied or not.
- The Set PCI Hardware Interrupt function applies to all hot-plug slots whether they are occupied or not.

### 5.2.2. Allocation of System Resources

In the context of this section, system resources are I/O Space addresses, Memory Space addresses, and bus numbers.

#### *5.2.2.1. System Resource Over-Allocation*

System resources are said to be over-allocated if more resources are allocated to each bus segment than are required to support the devices on that bus segment. Over-allocating resources to a hot-plug segment allows a limited number of new add-in cards to be hot-added without adjusting the resources allocated to the rest of the system.

Some operating systems never modify the system resource allocation established by the firmware at system boot. Firmware intended for use with such operating systems must over-allocate resources to each hot-plug bus segment if add-in cards are to be added to the bus segment after operating system initialization.

Other operating systems are able to change the system resource allocation that the firmware established at system boot. If an add-in card is hot-inserted in the system, such an operating system is able to reallocate system resources as required to provide resources for the new add-in card. It is optional for firmware to over-allocate system resources if it is intended for use with operating systems that reallocate system resources.

#### *5.2.2.2. Describing the Allocation of System Resources*

Regardless of whether system resources are over-allocated or not, information must be provided to the operating system to describe the allocation of resources as established by firmware.

DIG64 1.0 compliant Platforms must use ACPI to describe resource allocation. Examples of using ACPI to represent resource allocation are found in Section 5.5.

x86 Platforms must use one or both of the following methods to describe resource allocation. If the firmware is intended for use exclusively with operating systems that require only one method, the firmware is permitted to implement only that one method. If the firmware is intended to work with any operating system, the firmware is required to support both of the following methods:

- Use ACPI 1.0b or 2.0 to describe the allocated resources. (See Section 5.5.)
- Use the hot-plug resource table (HPRT) defined in Section 5.4 to describe the allocated resources.

### 5.2.3. Host Bridges

Firmware provides the base addresses of the Host Bridge Register Blocks of host-bridges with integrated SHPCs.

DIG64 1.0 compliant Platforms must provide this information using ACPI.

x86 platforms must use one or both of the following methods to provide base address information on host-bridges with integrated SHPCs. If the firmware is intended for use exclusively with operating systems that require only one method, the firmware is permitted to implement only that one method. If the firmware is intended to work with any operating system, the firmware is required to support both of the following methods:

- Use ACPI to describe host-bridges with integrated SHPCs. (See Section 5.5.3.)
- Use the HPRT to provide the base address of host-bridges with integrated SHPCs. (See Section 5.4.2.)

## 5.3. Hibernation

In this specification, the term “hibernation” is used to describe the mechanism by which the operating context of the system is saved to non-volatile media and the system is powered down. When the Platform is powered up after hibernation, the system loads the operating context that was previously saved.

Traditionally, the BIOS or firmware handles saving and restoring the operating context. In addition, ACPI 1.0b and ACPI 2.0 define a hibernation state called S4BIOS. S4BIOS is similar to BIOS hibernation. In this document, both of these hibernation states are referred to as firmware-controlled hibernation.

For firmware-controlled hibernation, the system firmware saves the state of the system on non-volatile media and powers-down the system. At the next system power-up, firmware restores the state of the system and continues to resume the operating system.

The presence of an SHPC in the system adds some additional requirements to system firmware in the implementation of hibernation. For S4BIOS, this is in addition to what is specified in ACPI 1.0b and 2.0.

When devices are hot-added, the system configuration changes with respect to its state at startup. In the case of an operating system that is capable of reallocating resources, the resources that are assigned by the operating system to the new add-in cards is, in most cases, very different from how the firmware originally assigned them when the system was first powered on. When the system hibernates and then resumes from that state, the firmware cannot allocate resources to the add-in cards the way it normally would at system power-up. If it did so, the resulting allocation of system resources would be different from that which existed before the system entered hibernation, which would

make it impossible for the operating system drivers to access their devices. Instead, the firmware must preserve the resource allocations that exist when the system enters hibernation and restore these allocations when the system resumes. This is in addition to the other hardware state information that is normally saved in hibernation.

The BIOS must save and restore all registers that affect the configuration and allocation of resources for PCI. This includes, but is not limited to, bus numbering, bridge apertures, memory resource allocations, I/O resource allocations, interrupt numbers, command registers, cache line sizes, and latency timers. Also the SHPC registers that must be initialized after resuming from hibernation are described in Section 5.6.

When a hot-plug system resumes from hibernation, if the firmware detects that a card has been added, changed, or removed during S4BIOS, the firmware must halt with an error message. The user must restore the hardware to its state prior to hibernation before firmware can allow the restoration from hibernation to proceed.

### Implementation Note: S4 Operating System Controlled Hibernation

ACPI 1.0b and ACPI 2.0 also define another hibernation state called S4. To enter the S4 state, the operating system (rather than the firmware) is required to save the machine state and power down. During the next power up, the system resumes from hibernation and the operating system restores the saved machine state. This procedure depends on the hardware configuration remaining unchanged between the time when the system goes into hibernation and when it resumes from hibernation.

The ACPI 1.0b and 2.0 specifications include a DWORD variable called a Hardware Signature in the Firmware ACPI Control Structure (see Table 5-11 in Section 5.2.9 in ACPI 2.0). Its purpose is to detect hardware changes. When the operating system hibernates, it saves the current Hardware Signature. When resuming from hibernation, the operating system compares the new Hardware Signature and the saved one. If the signatures are the same, resume from hibernation continues. If the signatures are not the same, resume from hibernation is not guaranteed to work.

Hot-plugging add-in cards introduces two additional problems for systems that implement S4 hibernation:

- The original Hardware Signature is computed when the system initially boots. However, it is not recomputed when a card is hot-added. Therefore, if a card is hot-added before the system goes into S4 hibernation, the Hardware Signature calculated when the system resumes from hibernation will not match the original one. This results in the failure to successfully restore a hibernated state even though the state of the hardware did not change while the system was hibernating.
- During system operation, as add-in cards are added and removed, it is possible for the system to have the same cards in the same slots as when the system first booted, but using different resources (for example, a different address range in Memory Space). When the system resumes from hibernation, firmware configures the PCI devices and assigns system resources according to its own algorithms, without any knowledge of how resources were assigned before the system entered hibernation. In this case, the Hardware Signature remains the same since the hardware configuration is identical; however, the resume will fail because of the different system resource assignments.

PCI hot-plug systems that implement S4 hibernation must provide system-specific solutions (that can include firmware) to these problems or risk not being able to resume from S4 hibernation.

# 5.4. Hot-plug Resource Table (HPRT)

The HPRT describes general system configuration and host-bridge resource allocations. Some operating systems require this information, as described in Sections 5.2.2.2 and 5.2.3, and use it when performing hot-plug operations. The HPRT only exists on x86 Platforms. The table is located on a 16 byte boundary and is found by searching the F000h segment for the "$HST" signature string.

The absence of a hot-plug resource table indicates either that ACPI is the primary method of providing information about hot-plug resource allocations and host-bridges with integrated SHPC or the absence of a hot-plug subsystem.

## 5.4.1. HPRT Structure

The HPRT structure optionally appears monolithically or is divided into two parts. If space is limited within the firmware memory, the second part of the HPRT structure is permitted to be located anywhere in memory. Figure 5-1 shows a monolithic HPRT structure. Figure 5-2 shows the HPRT structure that is in two parts. A 64-bit pointer points to the second part of the HPRT.

| 31 24 | 23 16 | 15 14 | 0 | Byte Offset |
|---|---|---|---|---|
| Signature<br>T | S | H | $ | 00h |
| Table Major Version | Table Minor Version | Flag = 0 | Number of Host Bridge Entries | 04h |
| Latency Timer | Cache Line Size | IRQ Bitmap Unused | | 08h |
| Host Bridge Entry 1 | | | | 0Ch |
| Host Bridge Entry 2 | | | | |
| . . . | | | | . . . |
| Host Bridge Entry N | | | | . . . |

**Figure 5-1: Monolithic HPRT Structure**

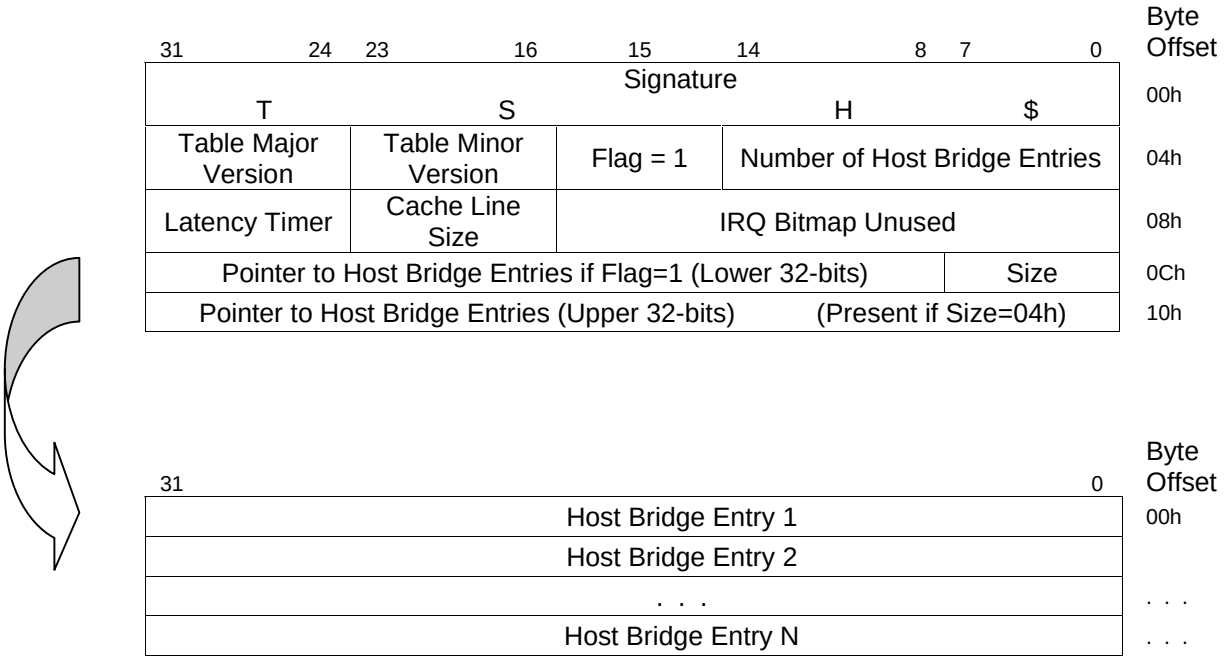



**Figure 5-2: HPRT Structure in Two Parts**

### Signature

The signature consists of the string "$HST". Software performs a scan of the F000h segment searching for this string on a 16-byte boundary. Once the signature is found, the Tbl Major Version and Tbl Minor Version are checked to determine whether they have a legal value as described below.

### Tbl Major Version

The Tbl Major Version is defined to be 01h for this specification. All other values are reserved. Future versions of this specification will use this field along with the Tbl Minor Version to determine the format of the table.

### Tbl Minor Version

The Tbl Minor Version is defined to be 00h for this specification. All other values are reserved. Future versions of this specification will use this field along with the Tbl Major Version to determine the format of the table.

### Flag

If 1, the host bridge entries are not consecutive to the table and the Pointer to Host Bridge Entries points to the actual table of entries in memory. If 0, Host Bridge Entry 1 follows immediately after the DWORD containing the IRQ Bitmap Unused.

### Number of Host Bridge Entries

This field describes how many Host Bridge Entries are contained within the table.

### IRQ Bitmap Unused

Thisis a bitmap that indicates which of the standard AT IRQs have not been allocated by the firmware for use by the system. A bit is 1 to indicate an unallocated IRQ. Bit

position *n* corresponds to IRQ*n*, where the least significant bit is bit 0 and the most significant bit is bit 15. For example, if IRQ 14 is not assigned, then bit 14 of this 16-bit field would be set.

**Cache Line Size**

The cache line size value to be written into the Cache Line Size register of hot-plugged devices.

**Latency Timer**

The latency timer value to be written into the Latency Timer register of hot-plugged devices.

**Pointer to Host Bridge Entries**

This field exists if Flag is 1. It points to the physical address of the Host Bridge Entries table in memory. This field is 32 or 64 bits long depending on the value of Size.

**Size**

If 00h, the Pointer to Host Bridge Entries is 32 bits long. If 04h, the Pointer to Host Bridge Entry is 64 bits long. All other values are reserved.

**Host Bridge Entry 1 … N**

Each Host Bridge Entry describes resource information pertaining to one of the Platform host bridges as specified in Section 5.4.2. All host bridges in the Platform require a Host Bridge Entry, regardless of whether or not they include an SHPC. The beginning of each Host Bridge Entry immediately follows the end of the previous Host Bridge Entry, even though Host Bridge Entries vary in size.

## 5.4.2. Host Bridge Entry

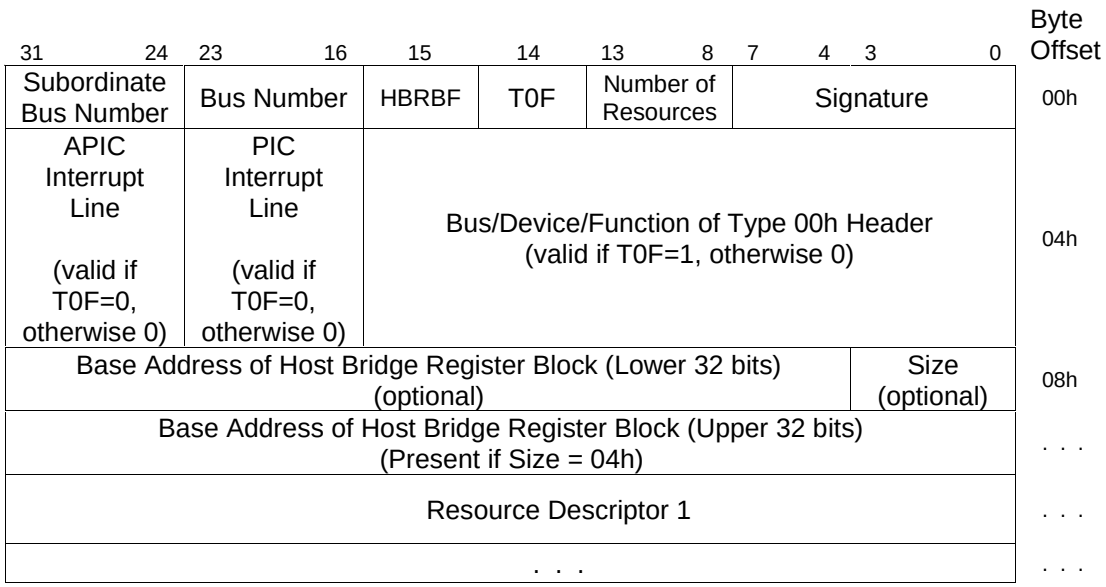



**Figure 5-3: Host Bridge Entry**

**Signature**

The signature is defined to be 19h. This allows software to validate the start of a host bridge entry.

**Number of Resources**

The number of resource descriptors for this host bridge entry.

**T0F**

Type 00h Header presence flag. If 1, this bit indicates that the 16 bit word at offset 04h is valid and indicates the location of the Type 00h header of the host bridge in Configuration Space. If this bit is 0, the APIC Interrupt Line and PIC Interrupt Line fields at offset 04h are valid.

**Host Bridge Register Block Flag (HBRBF)**

If 1, this bit indicates the presence of a host bridge with an integrated SHPC. The entry at offset 08h contains the address of the Host Bridge Register Block for the host bridge.

**Bus Number**

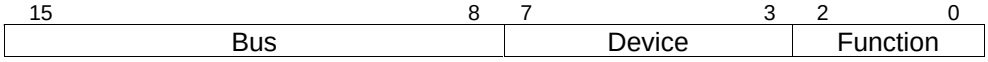
This field specifies the bus number of the PCI bus created by the bridge.

**Subordinate Bus Number**

This field specifies the number of the highest number hierarchical bus that is subordinate to the bridge in the PCI bus hierarchy.

**Bus/Device/Function of Type 00h Header**

If T0F=1, this word contains the bus/device/function number of the Type 00h header of the host bridge. The word is encoded as follows:




**APIC Interrupt Line and PCI Interrupt Line**

These fields are optional. They are present if T0F=0, and specify the interrupt line numbers that are used by the SHPC for both APIC and PIC modes of the Platform. Otherwise, the operating system obtains the SHPC interrupt number from the Interrupt Line register in the Configuration Space header of the indicated Bus/Device/Function.

**Base Address of Host Bridge Register Block**

Physical address of the Host Bridge Register Block. This field is optional. It is present if the HBRBF bit is 1. This field is 32 or 64 bits long depending on the value of Size. If not present, Resource Descriptor fields begin at the third DWORD of the Host Bridge Entry.

**Size**

If 0, the Base Address of the Host Bridge Register Block is 32 bits long. If 04h, the Base Address of the Host Bridge Register Block is 64 bits long. All other values are reserved. This field is optional. It is present if the HBRBF bit is 1.

**Resource Descriptors 1 … N**

Each resource descriptor field describes the range of I/O or memory addresses allocated to the bridge. The overall number of descriptors is indicated by the Number of Resources field. The size of the entries in the resource descriptors is either 32 bits or 64 bits as described in the following section.

## 5.4.3. Resource Descriptor

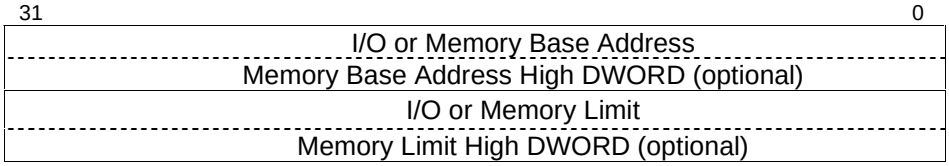



**Figure 5-4: Resource Descriptor**

Each host bridge entry describes an arbitrary number of I/O and memory apertures. An aperture is an address space resource that is allocated to the bridge. Two I/O or memory address values are necessary to describe each resource.

**I/O or Memory Base Address**

This field describes the lowest address of the resource. It is either a 32-bit I/O, or 32-bit memory, or 64-bit memory address. The four least significant bits are similar to those of a PCI Base Address register (as defined in PCI 2.2, Section 6.2.5.1) and indicate the actual size and type of the address as defined in Figure 5-5 and Figure 5-6. These figures show a 32-bit Base Address register or the lower 32-bits of a 64-bit Base Address register.

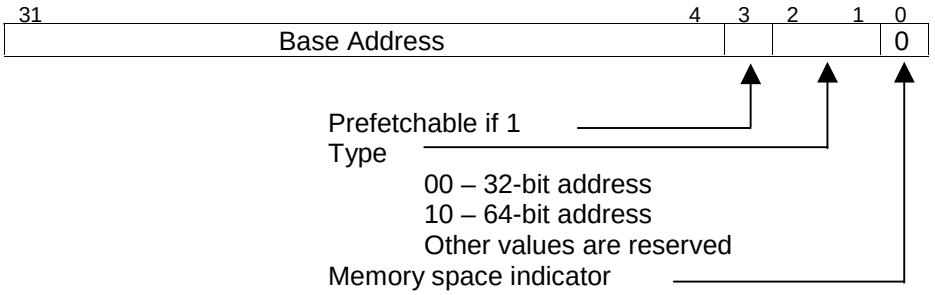



**Figure 5-5: Base Address for Memory Resource**

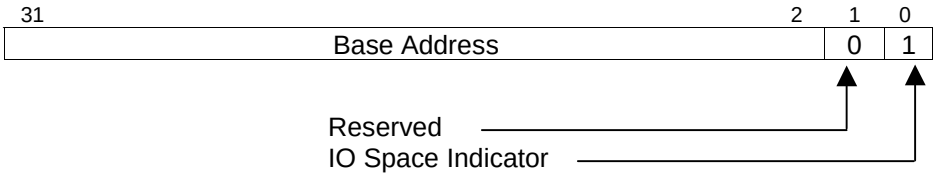



**Figure 5-6: Base Address for I/O Resource**

**I/O or Memory Limit**

This field describes the highest address of the resource. It is either a 32-bit I/O, or 32-bit memory, or 64-bit memory address. The least significant bits of the corresponding Base

Address field in Figure 5-5 and Figure 5-6 indicate the actual size and type of the address.

## 5.5. ACPI Firmware

The Advanced Configuration and Power Interface is a firmware standard used on x86 and DIG64 1.0 compliant Platforms. (See ACPI 1.0b/2.0.)

The firmware requirements are different depending on the type of bridge integrated with the SHPC and are detailed in the sections below.

The ACPI Source Language (ASL) examples below illustrate only the SHPC-related entries and are not complete descriptions for these devices. Refer to the ACPI specification for more information.

Table 5-1 summarizes the control methods that are provided for each form of SHPC.

**Table 5-1: Summary of ACPI Control Method Requirements**

| Control Method | PCI-to-PCI Bridge | Host Bridge |
|---|---|---|
| *_HPP* | (Note 1) | (Note 1) |
| *_PRW* | No | (Note 2) |
| *_PSW* | No | (Note 2) |
| *_PSx* | No | Optional |
| *_Lxx* | No | (Note 2) |
| *HBRB* | No | Required |
| *OSHP* | Required if firmware controls SHPC | Required if firmware controls SHPC |

Notes:
1. Required for ACPI 2.0 and recommended for ACPI 1.0b
2. Required if the system supports sleep states and thus uses **WAKE#** (see Section 6.2.2).

### 5.5.1. The *OSHP* Method

Some systems that include SHPCs that are released before ACPI-compliant operating systems with native SHPC support are available, use ACPI firmware to operate the SHPC. Firmware control of the SHPC must be disabled if an operating system with native support is used. Platforms that provide ACPI firmware to operate the SHPC must also provide a control method to transfer control to the operating system. This method is called *OSHP* (Operating System Hot Plug) and is provided for each SHPC that is controlled by ACPI firmware.

Operating systems with native SHPC support must execute the *OSHP* method, if present, for each SHPC before accessing the SHPC's registers and when returning from a hibernated state. If an SHPC's *OSHP* method is executed multiple times, and the switch to operating system control has already been achieved, the method must return successfully without doing anything. After the *OSHP* method is executed, the firmware must not access the SHPC registers. If any signals such as the System Interrupt or **PME#** have been redirected for servicing by the firmware, they must be restored appropriately for operating system control.

The following is an example of a namespace entry for an SHPC that is managed by firmware.

```
Device(PPB1){

      ...

      Method(OSHP, 0) {
            // Disable firmware access to SHPC and  restore
            // the normal System Interrupt and Wakeup Signal
            // connection.   (See the Implementation Note on page 157.)
      }
      ...
}
```

**Implementation Note: Controlling the SHPC by Using ACPI**

When using ACPI to control the SHPC, the following should be considered:

Firmware should redirect the System Interrupt and the Wakeup Signal to a GPE so that APCI can service the interrupts instead of the operating system. An appropriate *_Lxx* GPE handler should be provided. When an operating system with native SHPC support executes the *OSHP* method, the firmware restores the normal System Interrupt and Wakeup Signal connection so the interrupts can be serviced by the operating system.

In a PCI-to-PCI bridge implementation, access to the SHPC registers is recommended to be done through the DWORD Select/DWORD Data pair in Configuration Space and not the memory Base Address register. This is because versions of ACPI prior to 2.0 do not allow memory access through a Base Address register. In a Host Bridge implementation, the Host Bridge Register Block can be accessed directly by declaring a memory space operation region to encompass it. For more details, refer to the paper *Hot Plug PCI on Windows Operating Systems* available from Microsoft's web site.

### 5.5.2. The *_HPP* Method

The *_HPP* method is defined in Section 6.2.5 of ACPI 2.0. For every hot-plug slot that is controlled by an SHPC, a Platform that uses ACPI 2.0 firmware must provide the *_HPP* method for at least one bridge that is higher than the slot in the PCI bus hierarchy in Configuration Space. Platforms that use ACPI 1.0b are recommended to do the same.

### 5.5.3. SHPC Integrated with a Host Bridge

An SHPC integrated with a host bridge must be described in the ACPI device representing the host bridge. This ACPI device must have a hardware ID (*_HID*) of "PNP0A07" and a compatible ID (*_CID*) of "PNP0A03."

The firmware must describe where to find the Host Bridge Register Block for each host bridge, as defined in Section 4.3.2.1. The *HBRB* (Host Bridge Register Block) control method describes the base memory address of the header.

The memory range that is used by the Host Bridge Register Block must be included in the *_CRS* method (and all alternative configurations of the *_PRS* method, if present) for the host bridge with which the SHPC is integrated. Because the Host Bridge Register Block is used for configuration purposes, it must be available whenever the host bridge is present, for example, when the firmware first accesses a newly hot-inserted host bridge. (Note that hot-inserting a host bridge that is integrated with an SHPC is different from hot-inserting an add-in card using the SHPC. The details of hot-inserting a host bridge are beyond the scope of this specification.) This availability requirement imposes the following restrictions on memory occupied by the Host Bridge Register Block:

- The Host Bridge Register Block must be in a fixed location that never changes when the host bridge is physically present in the Platform and available to the software. Changing the resource settings of the host bridge by running the *_SRS* method must not change the location of the memory region decoded by the Host Bridge Register Block.
- The Host Bridge Register Block for a host bridge when it is newly inserted and when the system is first powered-up must decode its memory range as soon as the *_STA* method for the host bridge indicates the presence of the device. It must not rely on the prior assignment of resources to the host bridge or the loading of a device driver to be able to decode its memory range.

To be backward compatible with previous ACPI compliant operating systems (which do not evaluate the *HBRB* method), these resources must be claimed by an ACPI device with a hardware identification (*_HID*) of "PNP0C02" (Reserved Motherboard Resources) that is a child of the host bridge device in the namespace. This will prevent ACPI-compliant operating systems without native SHPC support from configuring a device in conflict with the Host Bridge Register Block. An ACPI compliant operating system that includes native SHPC support must evaluate the *HBRB* method and use this information to address any apparent conflicts between this "PNP0C02" device and the Host Bridge Register Block.

The SHPC also generates System Interrupts. In a host bridge, the interrupt pin associated with the SHPC is included in the *_CRS* method (and all alternative configurations of the *_PRS* method, if present). It is required that an active low, level triggered interrupt be used for this interrupt. The interrupt associated with the SHPC is the only interrupt permitted to be included in the *_CRS* method (and all alternative configurations of the *_PRS* method, if present). If the bridge includes a source of interrupts other than the SHPC, all interrupt sources (including the SHPC) must be either routed to a single interrupt signal, as in the case of a PCI-to-PCI bridge, or declared as separate ACPI namespace entries that are peers of the host bridge.

#### 5.5.3.1. HBRB *Control Method*

This control method provides the 64-bit physical address of the Host Bridge Register Block described in Section 4.3.2.1. When evaluated, it returns a package containing two 32-bit integers. The first contains the lower 32 bits of the address and the second the upper 32 bits.

```
Name(HBRB, Package(){
                        0xffd00000, // Bits 31:0
                        0x00000000  // Bits 64:32
                  })
```

#### 5.5.3.2. *Host Bridge Power Management*

Host bridge power management is not described by PCI PM 1.1 and, therefore, the wakeup requirement and any optional power management features must be provided by firmware.

In order to support the wakeup requirements of Section 6.2.2, the **WAKE#** signal, if implemented, must be connected to a GPE and this connection must be described by *_PRW*, *_PRS* methods and an appropriate *_Lxx* GPE handler method. It is recommended that an *_HPP* method be used to provide the operating system with extra configuration information for cards added to the system after initial boot. Optionally, *_PS0*, *_PS1*, *_PS2*, and/or *_PS3* methods provide for host bridge power management.

## Implementation Note: WAKE# Routing

The **WAKE#** signal is used for host bridges in the same way as **PME#** is used for PCI-to-PCI bridges. It is connected to a GPE which can be programmed by the operating system to wake the machine from a sleeping state when **WAKE#** is asserted.

The GPE to which **WAKE#** is connected is indicated by the *_PRW* method. The *_PRW* method is hierarchical. That is, a bridge can only have one *_PRW* method, and it also indicates where the **PME#** signals for all subordinate PCI devices are connected (unless they are subordinate to an additional bridge with its own *_PRW* method). **WAKE#** enabling and disabling is controlled by the *_PSW* method, which is also hierarchical. Therefore, the **PME#** signals for all subordinate devices included in the host bridge's *_PRW* method must be gated by the *_PSW* switch in the host bridge.

ACPI also requires that only **PME#** signals from the same PCI bus hierarchy be connected to the same GPE.

Figure 5-7 illustrates a legal routing of **WAKE#** and **PME#** signals for ACPI applications.

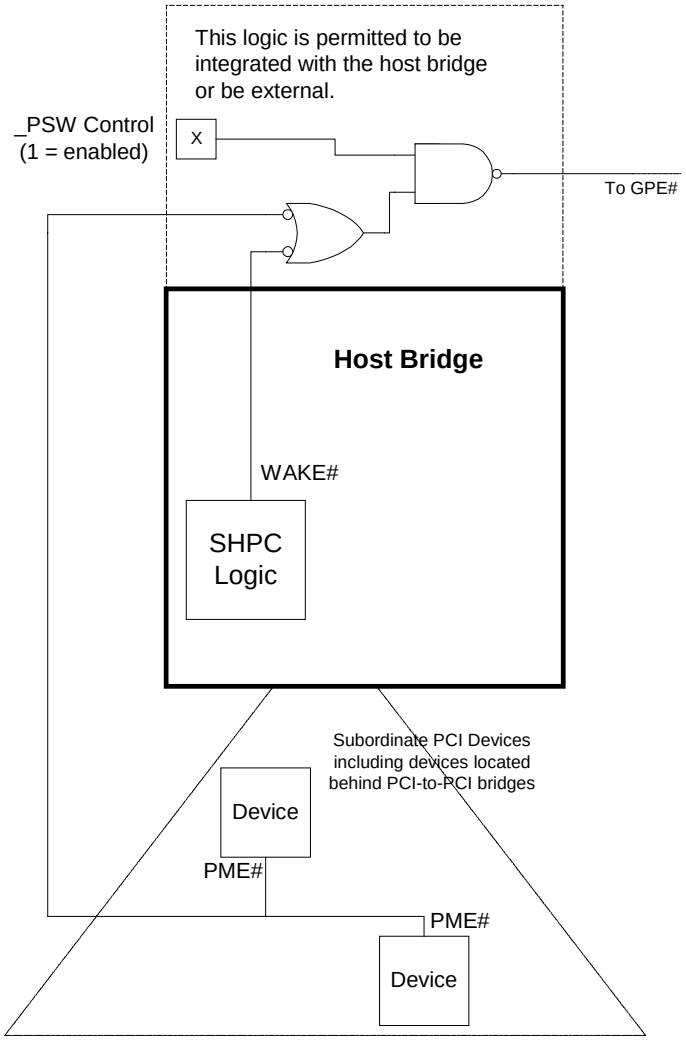



**Figure 5-7: WAKE# and PME# Routing in Host Bridges**

For more information, refer to the paper *GPE Routing for Microsoft Windows*, available from Microsoft's web site.

## 5.5.4. Example Namespace

### *5.5.4.1. SHPC Integrated with a PCI-to-PCI Bridge*

The following is an example of a namespace entry for an SHPC that is integrated with a PCI-to-PCI bridge that resides at device 0Eh, function 0:

```
Device(PPB1){

      ...

      //
      // _ADR is required by ACPI to identify the device.
      // In this case Device E Function 0
      //

      Name(_ADR, 0x000e0000)

      //
      // _HPP tells OS how to configure
      // parameters normally configured by BIOS during POST
      //

      Name(_HPP, Package(){
                        0x08, // CacheLineSize in DWORDS
                        0x40, // LatencyTimer in PCI clocks
                        0x01, // Enable SERR (Boolean)
                        0x00  // Enable PERR (Boolean)
                        })

      //
      // OSHP Method if required
      //
      Method(OSHP, ...)
      ...

}
```

### *5.5.4.2. SHPC Integrated with a Host Bridge*

The following is an example of a namespace entry for a host bridge integrated with an SHPC:

```
Device(PCI0){
      //
      // Recommended _HPP which tells OS how to configure
      // parameters normally configured by BIOS during POST
      //

      Name(_HPP, Package(){
                        0x08, // CacheLineSize in DWORDS
                        0x40, // LatencyTimer in PCI clocks
                        0x01, // Enable SERR (Boolean)
                        0x00  // Enable PERR (Boolean)
                        })

      //
```

```
// _PRW and _PSW methods to control system wakeup.
// These methods are required if device power management is
// supported.
//

Name(_PRW, Package(){
                  0x3, // WAKE# is wired to GPE bit 3
                  0x4  // We can wake from S4
                  })

Method(_PSW, 1) {
      if (Lequal(Arg0,0) {
            // Disable WAKE# assertion
      } else {
            // Enable WAKE# assertion
      }
}
//
// Optional _PSx methods to control power state
//

Method(_PS0) {...}      // Take Host Bridge to D0
Method(_PS1) {...}      // Take Host Bridge to D1
Method(_PS2) {...}      // Take Host Bridge to D2
Method(_PS3) {...}      // Take Host Bridge to D3

//
// OSHP Method if required
//
Method(OSHP, ...)



Name(_CRS,  ResourceTemplate(){
      //
      // IO space on the bus
      //
      IO(...),

      //
      // Bus numbers on this bus
      //
      WORDBusNumber(...),

      //
      // Memory space on the bus
      // N.B.  This is just a fictional example and is
      // probably not appropriate for your machine!
      //
      DWORDMemory(
            ResourceProducer, PosDecode,  MinFixed,
            MaxFixed,   Cacheable, ReadWrite,
            0x00000000, // Granularity
            0x08000000, // Min = Top of RAM (128 Mb)
            0xffcfffff, // Max = 4 GB - 2 MB - size of HBRB
            0x00000000, // Translation
            0xf7d00000, // Range Length 4 GB - 128 Mb RAM
            , ,         // Optional fields left blank
            )
      }
```

```
        //
        // Memory space used by the Host Bridge Register
        // Block.  This range must not change even if the
        // other resources decoded by this host bridge change
        // through a change in the _PRS alternative being
        // used.
        //
        DWORDMemory(
              ResourceConsumer, PosDecode, MinFixed,
              MaxFixed, Cacheable, ReadWrite,
              0x00000000, // Granularity
              0xffd00000, // Min
              0xffdfffff, // Max
              0x00000000, // Translation
              0x00100000, // Range Length
              , ,         // Optional fields left blank
              )

        //
        // The interrupt that will be asserted by the SHPC,
        // in this case interrupt 9
        //
        Interrupt(ResourceConsumer,...) {9}
)

//
// HBRB describes the 64-bit physical base address of the
// Host Bridge Register Block which is
// described in Section 4.3.2.1.  It is structured as a
// package of two 32-bit integers with the low order bits
// first.
//

Name(HBRB, Package(){
                        0xffd00000, // Bits 31:00
                        0x00000000  // Bits 64:32
                  }
)

//
// The resources associated with the HBRB
// registers are claimed by a PNP0C02 node for backward
// compatibility reasons.  The entire region decoded by the
// host bridge should be claimed and therefore the length
// of region should be the same as the Host Bridge Register
// Block Size register in the Host Bridge Register Block
// Header.
//
```

```
        Device(REGS) {
              Name(_HID, EISAID("PNP0C02"))

              Name(_CRS, ResourceTemplate(){
                    //
                    // Memory resources for the HBRB
                    //
                    Memory32Fixed(
                          ReadWrite,
                          0xffd00000,
                          0x00010000,
                          // Optional name omitted
                          )

              })
        ...
}

//
// The Required _Lxx handler for WAKE#.  In this case it is
// connected to GPE 3.
//

Scope(\_GPE){
        Method(_L03){
              // Clear WAKE# Status and thus stop asserting WAKE#
        }
}
```

## 5.6. State of the SHPC Initialized by the Firmware

Table 5-2 shows the state of the SHPC at the end of hot-plug subsystem initialization as a result of a power-on, software-initiated reset or a resume from a low power state.

**Table 5-2: State of the SHPC Initialized by the Firmware**

| Register Bit | After Boot and After Resume From Hibernate/Low Power State | Comments |
|---|---|---|
| Slots Available Registers I and II | Set to Platform-specific configuration for bus speed/mode capabilities | (Note 1) |
| Number of Slots Implemented | Set to number of physical slots | (Note 1) |
| First Device ID | PCI **DEVSEL#** number assigned to first slot in the bus segment | (Note 1) |
| Physical Slot Number Up/Down | Indicates the direction of enumeration of the slots in the bus segment | (Note 1) |
| MSL Sensors Implemented | Indicates whether the MRL is implemented | (Note 1) |
| Attention Button Implemented | Indicates whether the Attention Button is implemented | (Note 1) |
| Physical Slot Number | Label of the first slot in this bus segment | (Note 1) |
| Global Interrupt Mask | 1 | (Note 2) |
| Arbiter Timeout SERR Mask | 1 | (Note 2) |
| Command Complete Interrupt Mask | 1 | (Note 2) |
| Card Presence Interrupt Mask | 1 | (Note 2) |
| Attention Button Interrupt Mask | 1 | (Note 2) |
| MRL Sensor Interrupt Mask | 1 | (Note 2) |
| Isolated Power Fault Interrupt Mask | 1 | (Note 2) |
| MRL Sensor SERR Mask | 1 | (Note 2) |
| Non-isolated Power Fault SERR Mask | 1 | (Note 2) |
| Non-isolated Power Fault Interrupt Mask | 1 | (Note 2) |
| Slot Event Latch field | (Note 3) | |

Notes:
1. Set by firmware if a hardware-only method is not used. (See Section 3.5.1.4.)
2. This is the default state of the hardware after primary reset and the firmware does not change it.
3. Events that occur during boot (after firmware has initialized the SHPC but before the operating system has loaded) are reported in the slot event latch register.

# 6. Power Management of the SHPC

## 6.1. Introduction

All the requirements described in PCI Bridge 1.1 and PCI PM 1.1 related to PCI-to-PCI or host bridges also apply to bridges with integrated SHPCs. This chapter describes the additional power management capabilities introduced by integrating an SHPC and bridge and how they are used to implement the Standard Usage Model described in Section 2.6.

This chapter uses power management terminology from PCI PM 1.1 (for example, ***D0***, ***D1***, and so on for device power states and ***B0***, ***B1***, and so on for bus power states). Since PCI PM 1.1 does not define some terms for host bridges, this chapter uses the following terminology to generalize the mechanisms used by both PCI-to-PCI bridges and host bridges to wake the system or the SHPC from a low power state:

- Wakeup Signal – Wakeup Signal is used to refer generically to either the **PME#** signal from a PCI-to-PCI bridge, as defined in Chapter 7 of PCI PM 1.1, or the **WAKE#** signal from a host bridge, as defined in Section 6.2.2. Although Wakeup Signal refers to both **PME#** and **WAKE#**, they are different signals. Routing of **WAKE#** is described in Section 5.5.3.2.
- Wakeup Context – For a PCI-to-PCI bridge, the Wakeup Context is the PME Context, defined in Section 1.4 of PCI PM 1.1. Wakeup Context for a host bridge consists of the host bridge's equivalent of the PME_En bit, PME_Status bit plus any device class specific context. The device class specific context for SHPCs is defined in Section 6.2.3 below.
- Wakeup Status – Wakeup Status is used to refer generically to either the PME_Status bit in a PCI-to-PCI bridge or an equivalent bit in a host bridge.
- Wakeup Enable – Wakeup Enable is used to refer generically to either the PME_En bit in a PCI-to-PCI bridge or an equivalent bit in a host bridge.

## 6.2. Power Management Requirements

The SHPC Working Register set is accessible only when the bridge is in ***D0***. This means that commands can only be issued to the SHPC when it is in ***D0***. Note that while not accessible in low power states, the Wakeup Context is preserved in all power states as described in Section 6.2.3.

### 6.2.1. PCI-to-PCI Bridges with Integrated SHPCs

A PCI-to-PCI bridge integrated with an SHPC is required to provide a Power Management Capability List Item as described in Section 3.2 of PCI PM 1.1. Therefore, the power management state of the bridge is always the same as the power management state of the SHPC. As defined in PCI PM 1.1, the ***B*** states of the secondary bus are tied to the ***D*** states of the bridge. To allow support for the full range of secondary bus power states (***B0***, ***B1***, ***B2***, and ***B3***), the bridge is required to support device power states ***D0***, ***D1***, ***D2***, and $D3_{hot}$. The mapping of the ***D3*⇒*B3*** state is optional, as described in Section 3.2.5 of PCI PM 1.1. Support for $D3_{cold}$ is optional. If $D3_{cold}$ is supported, the Platform must provide **3.3Vaux** to support the wakeup requirements described below and to support the Standard Usage Model in Section 2.6.

A PCI-to-PCI bridge integrated with an SHPC is required to be capable of asserting **PME#** and must, therefore, implement a **PME#** pin and the PME_En and PME_Status bits as described in Section 7 of PCI PM 1.1. The bridge optionally requires that the PCI clock be present to support **PME#** generation and indicates this by setting the PME_Clock bit, as described in Section 3.2.3 of the PCI PM 1.1.

The SHPC Capability List Item in the PCI-to-PCI bridge must be accessible in all power states in which Configuration Space access is allowed (that is, all states except $D3_{cold}$). In all states other than ***D0***, accesses to the DWORD Select and DWORD Data registers are undefined.

### 6.2.2. Host Bridges with Integrated SHPCs

It is recommended that a host bridge that is integrated with an SHPC implement some form of power management. That is, such a host bridge is recommended to implement a method by which software places the host bridge or the PCI bus it creates in a reduced-power state. (This feature is optional, as described in Section 6.1 of PCI PM 1.1.) The power management state of the host bridge is always the same as the power management state of the SHPC. If the host bridge appears in Configuration Space, it must *not* include a standard PCI Power Management Capability List Item, because its meaning is undefined for a host bridge. If power management is provided in a host bridge, its detection and use is system specific. An example of an ACPI implementation is provided in Section 5.5.

If the system supports any sleep state, as defined in Section 2.6.1, the system is required to provide a wakeup mechanism to be compliant with the Standard Usage Model. The wakeup mechanism requires the implementation of a Wakeup Signal, **WAKE#**. The enabling and servicing of **WAKE#** is firmware specific but the hardware must provide Wakeup Status and Wakeup Enable bits to control the signal. An example of using ACPI to describe this is found in Section 5.5.

### 6.2.3. Wakeup Context

Wakeup Context for PCI-to-PCI bridges and for host bridges is defined in Section 6.1 above to include the Wakeup Status and Wakeup Enable bits, plus any device class specific context. The SHPC specific context is the Slot Event Latch field in the Logical Slot register for each implemented slot. (Although particular implementations are permitted to preserve additional state, system software assumes that only these registers are preserved.) The Wakeup Context is required to be preserved across the ***D3⇒D0*** internal reset and, in a bridge that supports the (optional) $D3_{cold}$ state, across hardware reset (see Section 3.5.1). This allows system software to determine the event that caused the Wakeup Signal to be asserted and to take the appropriate action once the system has awakened.

## 6.3. Event Signaling from Low Power States

When operating normally in ***D0***, system software configures the SHPC to report events using its System Interrupt or **SERR#** pin. When in a low power state, system software configures the SHPC to assert the Wakeup Signal instead of both generating a System Interrupt and asserting **SERR#**.

### 6.3.1. Controlling Wakeup Signal Assertion

When in low power states, the system is not prepared to acknowledge **SERR#** and System Interrupts. Therefore, software must program the SHPC so that the Wakeup Signal is asserted and wakes the system, returning it to a working state in which the System Interrupt or **SERR#** are re-enabled and processed. The enabling or disabling of **SERR#**, System Interrupt, and the Wakeup Signal when entering or leaving low power states is totally under software control. The SHPC hardware does not enforce any rules with respect to the power states, and the assertion of these signals operates as illustrated by Figure 4-24 and Figure 4-25.

Section 6.4 provides guidelines on using the enable and mask bits to comply with the Standard Usage Model.

### 6.3.2. Mapping SERR# Events to the Wakeup Signal

If any of the catastrophic events described in Section 4.7.3.2 occur while the SHPC is operating in ***D0***, the SHPC asserts **SERR#**. While in a low power state, the SHPC must not assert **SERR#**. Instead, the SHPC must be programmed to assert the Wakeup Signal as described for each of the events listed below.

#### *6.3.2.1. Unexpected MRL Opening*

When the secondary bus is operating in a state other than ***B0***, unexpected MRL opening does not risk data corruption because all devices in the secondary bus must be in ***D1*** or lower or have been quiesced, and are, therefore, idle.

However, since opening the MRL removes **3.3Vaux** from the slot, if the add-in card is using **3.3Vaux** to maintain its internal state, (such as PME Context) that state is lost. Loss of PME Context is not considered catastrophic. System software must continue to operate if PME Context is lost on a card where **PME#** assertion is enabled.

#### *6.3.2.2. Connected Power Fault*

A connected power fault (as defined in Section 3.4.2) that occurs while the SHPC is in ***D0*** and the system is in a working state (as defined by Table 2-15) potentially corrupts transactions actively running on the bus at the time of the fault. If system software is unable to determine the extent of the influence that such a fault has on the rest of the system or if system software cannot contain the impact of such a fault, system software sets the Slot SERR-INT Mask bits in the Logical Slot register to cause **SERR#** to assert immediately after a connected power fault to halt the possible spread of the corrupted data.

No traffic occurs on the bus while the SHPC is in a state other than ***D0*** because all devices on the secondary bus are quiesced. Although there is no risk of corrupting data on the bus, there is still a risk that a power fault that occurs on a single add-in card could corrupt the state of another device on the bus before the power fault is detected and the slot is disconnected by the hardware (see Section 3.1.1). Although it is not necessary in such cases to halt the system immediately, it is also not necessarily safe to attempt to resume execution.

**Implementation Note: Resuming Execution After a Connected Power Fault**

The design of the operating system determines whether or not software attempts to resume execution after a connected power fault in a sleeping state. The following issues should be considered:

- The impact on other devices on the bus (if there are any) is unpredictable. If there are other devices, resetting the secondary bus to return them to a consistent state is recommended.
- It may require a substantial time for the system software to resume from the low-power state. Therefore, during this time the effects of the power fault may become far reaching.
- Critical system resources, such as the system console or paging device, could be on the bus with the slot where the power fault occurred.
- Such a fault would cause a sleeping system to wake when the user expected it to be asleep. In this case, even if the power fault is handled gracefully the act of waking the machine may have been undesirable.
- Data on the add-in card with the power fault, or on other devices on the bus that are affected by the power fault, could be lost (for example, unflushed write cache).

## 6.4. Supporting the Standard Usage Model

This section describes the operations which system software must perform when changing the power state of the SHPC in order to support the Standard Usage Model for power managed systems described in Section 2.6.

### 6.4.1. Operations Performed When Entering a Low Power State

1. Disable generation of a System Interrupt by setting the Global Interrupt Mask bit in the Controller SERR-INT Mask register.
2. Disable the assertion of **SERR#** by setting the Global SERR Mask bit in the Controller SERR-INT Mask register.
3. Configure the Slot SERR-INT Mask field in each slot's Logical Slot register to select which events will cause the Wakeup Signal to be asserted. To comply with the Standard Usage Model, this must include clearing the Attention Button Interrupt Mask bit and setting the MRL Sensor SERR Mask bit.
4. Place the bridge into a low power state by writing to the PowerState register in the Power Management Capability List Item of a PCI-to-PCI bridge or by using the equivalent host bridge mechanism.
5. Enable assertion of the Wakeup Signal by setting the Wakeup Enable bit.

### 6.4.2. Operations Performed When Returning from a Low Power State

1. Disable and deassert the Wakeup Signal by clearing the Wakeup Enable and Wakeup Status bits.
2. Return the bridge to ***D0*** by writing to the PowerState register in the Power Management Capability List Item or by using the equivalent host bridge mechanism.
3. Configure the Slot SERR-INT Mask field in each slot's Logical Slot register to select which events will generate a System Interrupt or cause **SERR#** to be asserted.
4. Set the SERR# Enable bit in the PCI-to-PCI bridge's Command register, or equivalent host bridge register, to enable the assertion of **SERR#**. Then clear the Global SERR Mask bit in the Controller SERR-INT register. The SHPC generates the **SERR#** pulse, if there are any pending **SERR#** events. In some cases, this causes the system to halt execution.
5. Enable generation of a System Interrupt by clearing the Global Interrupt Mask bit in the Controller SERR-INT Mask register. Any pending interrupts that have not been masked in the SERR-INT Mask field of a Logical Slot register cause the SHPC to generate a System Interrupt at this point.

At this point, the system returns to a working state and any interrupts are delivered and serviced.

## 6.5. PME# and 3.3Vaux Control for Disabled Slots

Before system software disables a slot, it must clear PME_En in the Power Management Capability List Item of the device in the slot as part of the quiesce operation for that device. The Standard Usage Model requires that the slot's **PME{n}#** pin remain connected to the bus, even when the slot is disabled, until the MRL is opened.

# Appendix A. Switch Debounce Techniques

Most hot-plug Platforms use switches as a slot-side user interface to communicate the state of the MRL and Attention Button. In cost-sensitive applications, the economical choice is a mechanical switch (rather than a bounce-free optical or Hall-effect electronic switch). This appendix describes the contact bounce phenomenon and reliable techniques for debouncing mechanical switch contacts.

A mechanical switch/button virtually always exhibits an electrical bounce phenomenon. When a mechanical switch is opened or closed, the mechanical contacts do not break or make a connection instantaneously, but usually "bounce" between open and closed, thus making several transitions. Because electronic circuits (and software) are much faster than the bouncing behavior of a mechanical switch, they will sense multiple transitions (one to zero or zero to one) for a single mechanical transition. Typically, low-current (less than 1 A) switches will bounce for a period of 1 to 10 ms when the switch changes state. For electronic circuits to exhibit predictable behavior, the mechanical switch input must be “debounced” before being used by other electronic circuits that may ultimately control a state machine, generate a System Interrupt, or be presented to software in a readable register.

When selecting a mechanical switch for the low voltage applications (less than 5 V), the use of precious metal contact surfaces, such as gold, silver or various precious metal alloys is recommended. Otherwise, deterioration in contact resistance and bounce characteristics over the service life of the switch is likely.

Switches can be debounced in many ways. The most common debounce techniques rely on a time constant to perform time-domain filtering that maintains a constant state on the debounced switch output as long as the switch is bouncing. That is, the switch contacts (connected directly to the input of the debounce circuit) must maintain the same logic value for some minimum time period before the debounced output state is allowed to change. Time-domain filtering techniques can be implemented either in analog/digital hardware, digital hardware, or software. Another common debounce technique requires the use of a single-pole double-throw (SPDT) switch such that both the on and off contacts of the switch are used to perform the debounce function.

The circuit diagram in Figure A-1 depicts a simple single-pole single-throw (SPST) switch (S1) with one contact connected to signal ground and the other pulled up to $V_{DD}$ through resistor R1. The diagram includes two different types of gates so that the effects of switch bounce can be contrasted. Gate U1 is a conventional CMOS gate having a nominal threshold of $V_{DD}/2$. Gate U2 is a Schmitt-Trigger (hysteresis) gate whose switch input has been low-pass filtered by R2 and C1. The debounce circuit comprised of U2, R2, and C1 is an example of an analog/digital hardware implementation and is referred to as the “R/C+hysteresis gate debounce circuit” throughout the remainder of this discussion. The R/C filter smoothes out the bounce activity of the switch but the resulting signal has slow rise and fall times. When the rise/fall time of a signal exceeds the maximum rise/fall time specification of single-threshold gate, a hysteresis gate is required to insure monotonic output behavior. The hysteresis gate also plays a critical role in establishing the bounce immunity period.

As can be seen in the Figure A-1 timing diagram, the conventional gate reacts to every bounce of the switch contacts as the switch is opened and closed. The R/C filter integrates the bouncing contact to produce a smooth, slow-changing voltage signal (at node C) that ultimately crosses $V_{T+}$ of the hysteresis gate. At that time, the output of U2 (node D) changes from 0 to 1 in a clean digital fashion.

Furthermore, the R/C+hysteresis gate circuit has two timing characteristics of interest to the circuit designer. They are:

- Minimum Switch Latency ($T_{mSL}$)
- Maximum Switch Latency ($T_{MSL}$)

Switch latency is the amount of time from the last bounce event until the debounce circuit changes state. For a hysteresis gate, the minimum switch latency is the time required for the voltage on C1 (node C) to the cross from $V_{T-}$ to $V_{T+}$ or vice versa. The minimum switch latency parameter actually defines the circuit's bounce immunity. The minimum switch latency also determines the maximum frequency at which node D toggles. Given a worst-case bounce scenario, an opening switch will bounce in such a way that the voltage on C1 is slightly less than $V_{T-}$ before the switch becomes stable. After becoming stable, the voltage on C1 continues to rise to $V_{T+}$. $T_{mSL}$ is defined as the time required for a transition from $V_{T-}$ to $V_{T+}$ (or vice versa). At the instant the voltage on C1 crosses the $V_{T+}$ threshold, assume the switch returns to the closed position cleanly without a bounce. For this switch behavior, the debounced result (node D) would have a high state duration of $T_{mSL}$. Therefore, the maximum frequency of the debounced switch signal is $1/(2 * T_{mSL})$. However, this formula is only an approximation. Depending on how well the hysteresis gate thresholds are centered ($V_{T+} - V_{DD}/2 = V_{DD}/2 - V_{T-}$), gate leakage currents, switch pull-up voltage, and R1 pullup impedance, $T_{mSL}$ for an opening switch may not match $T_{mSL}$ for a closing switch.

The other timing parameter is defined as the maximum switch latency $T_{MSL}$. If the voltage on C1 is steady-state (equal to $V_{DD}$ or GND), $T_{MSL}$ is the time required for the debounced output to change after a single (non-bouncing) switch state transition. This behavior is illustrated in Figure A-2.

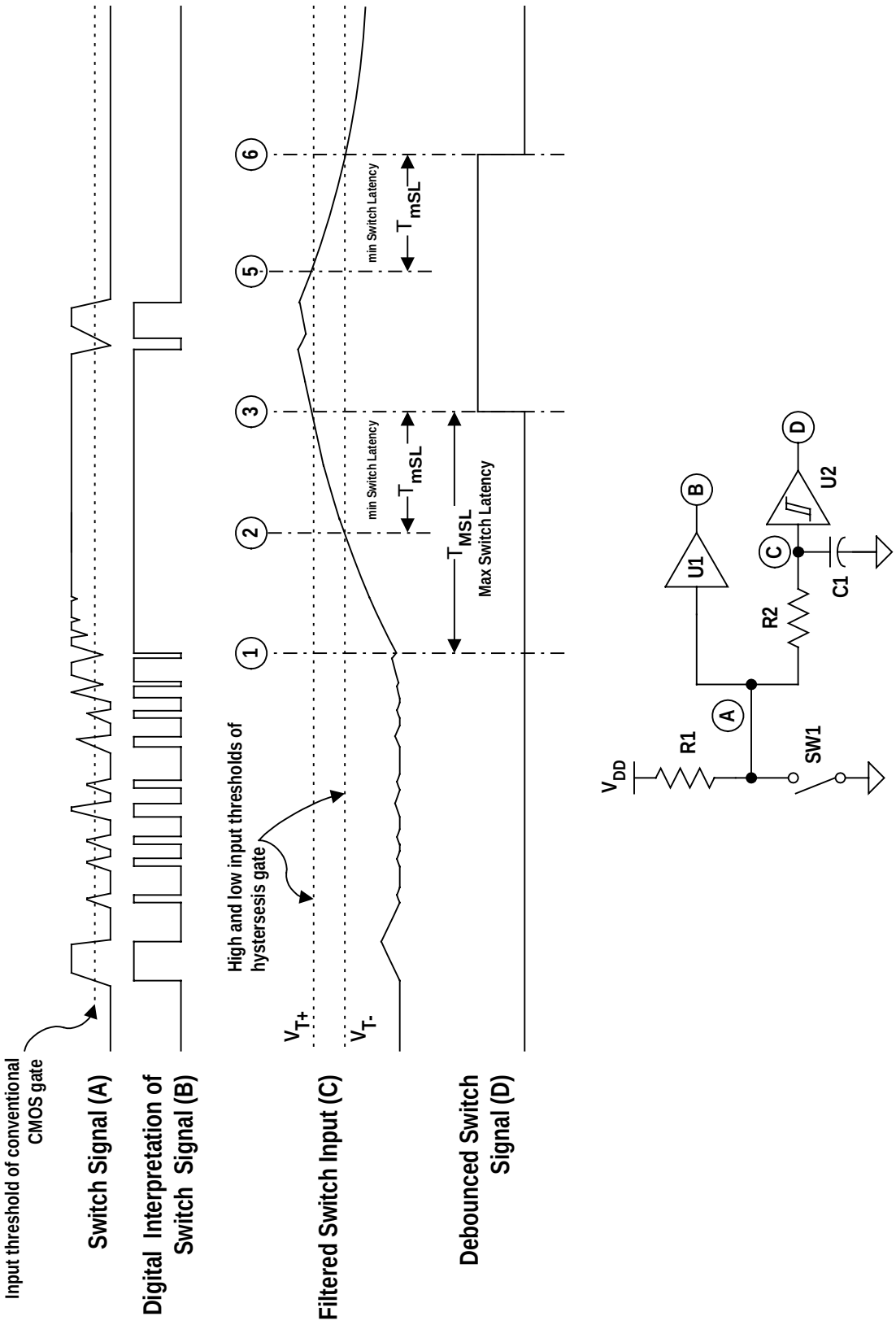



**Figure A-1: Mechanical Switch Bounce Behavior and R/C+Hysteresis Gate Debounce Circuit**

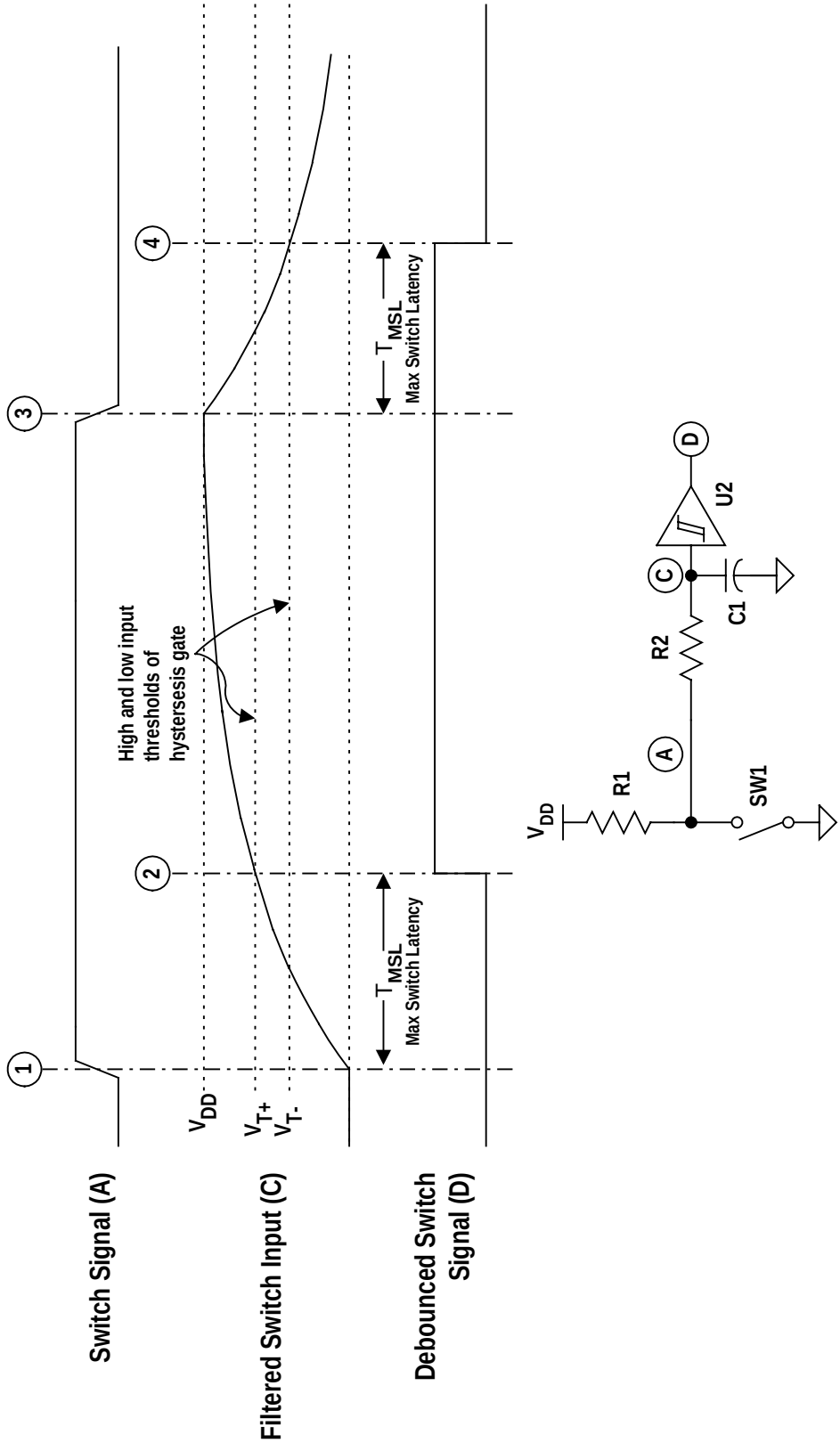



**Figure A-2: Maximum Switch Bounce Latency of an R/C+Hysteresis Gate Debounce Circuit**

Other design considerations for using R/C+hystersis gate debounce circuits are as follows:

- The value of R2 should be much greater than R1 so that the effective R/C time constant for opening and closing switch events are nearly the same.
- An R/C time constant (product of R2 and C1) that is significantly larger than the switch bounce period is usually required (see Table A-1).
- The largest value of R2 is dependent on the input leakage current characteristics of the hysteresis gate.
- The value for $V_{T-}$ to $V_{T+}$ should be equidistant from GND and $V_{DD}$ if $T_{mSL}$ must be the same for both closing and opening switch events.
- The values of $V_{T-}$ and $V_{T+}$ are a function of process, operating voltage, and temperature.
- Hysteresis gates from different vendors having the same part number could have radically different hysteresis characteristics.

**Table A-1: R/C+Hysteresis Gate Debounce Circuit Timing Characteristics**

| Gate | $V_{DD}$ | R2 | C1 (µF) | $V_{T-}$ | $V_{T+}$ | Hysteresis (volts) | $T_{mSL}$ (ms) | $T_{MSL}$ (ms) |
|---|---|---|---|---|---|---|---|---|
| 1 | 3.3 | 51000 | 2.2 | 1.575 | 1.725 | 0.150 | 10.207 | 82.991 |
| 2 | 3.3 | 51000 | 1.0 | 1.500 | 1.800 | 0.300 | 9.298 | 40.211 |
| 3 | 3.3 | 51000 | 1.0 | 1.400 | 1.900 | 0.500 | 15.574 | 43.730 |
| 4 | 3.3 | 51000 | 1.0 | 1.275 | 2.025 | 0.750 | 23.594 | 48.500 |
| 5 | 3.3 | 51000 | 1.0 | 1.150 | 2.150 | 1.000 | 31.911 | 53.762 |
| 6 | 3.3 | 51000 | 1.0 | 1.025 | 2.275 | 1.250 | 40.662 | 59.631 |
| 7 | 3.3 | 51000 | 1.0 | 0.900 | 2.400 | 1.500 | 50.022 | 66.263 |
| 8 | 3.3 | 51000 | 1.0 | 0.775 | 2.525 | 1.750 | 60.238 | 73.890 |
| 9 | 3.3 | 51000 | 1.0 | 0.650 | 2.650 | 2.000 | 71.672 | 82.860 |
| 10 | 3.3 | 51000 | 0.1 | 0.400 | 2.900 | 2.500 | 10.103 | 10.762 |

For example, to debounce a switch having a 10 ms bounce specification, an R/C time constant of 112.2 ms (the product of R2 and C1) is required when using a hysteresis gate with the characteristics of gate #1 shown in Table A-1. The circuit in Figure A-3 can be used to test the design and measure $T_{mSL}$. The circuit in Figure A-1 is modified so that the debounced switch signal (node D) is inverted by the CMOS gate, U1, that is powered by $V_{DD}$. When the inverter output is used to drive the node (node A) normally driven by the switch, the circuit will oscillate. $T_{mSL}$ can be measured using an oscilloscope. If the duty cycle of the signal is not 50%, the input current leakage of the gate is significant and the value of R2 should be decreased.

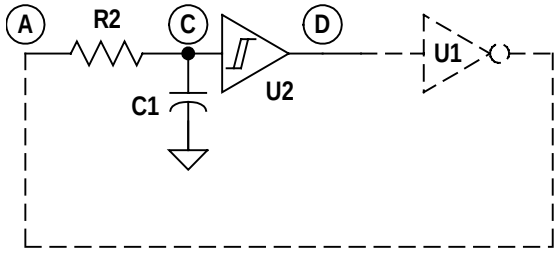



**Figure A-3: Minimum Switch Latency Measurement Circuit**

If highly predictable (less variability) switch latencies are required, a more complex circuit would be required that exhibits nearly equal values for $T_{MSL}$ and $T_{mSL}$. The debounce algorithm shown in Figure A-4 can be tuned to produce highly predictable switch latencies. The algorithm could be implemented in digital hardware (for example, a state machine in the SHPC) or as low-level software routine (for example, in a micro-controller used to collect slot-side inputs). As shown in Figure A-4, three 1-bit registers (Boolean variables) and one integer variable are used to perform the debounce function. The Boolean variable sw_input represents the bouncing switch data that is available as a hardware/software input. The variable could be continuously available as in a dedicated switch input pin on the SHPC or could be sampled and scanned using a serial interface. However, the algorithm assumes there is a wait period, $T_{wait}$, that represents the sampling period. The sw_state Boolean variable is the debounced switch output. The sw_sample Boolean variable is a temporary register that holds the value of sw_input last sampled. The timer is an integer variable that emulates the R/C low-pass filter of an analog circuit.

After initializing the timer to zero and setting both sw_state and sw_sample equal to the initial sampled value of sw_input, a wait is executed. Then a new sw_input sample is taken. If the value of the sw_input is *not equal* to that of the last sample (sw_sample), then the timer is initialized (or re-initialized) to an integer count value of $T_{mSL}/T_{wait}$. The sw_input value is then copied to sw_sample.

If the value of the sw_input is *equal* to that of the last sample, the timer is checked. If the timer is zero, no change has been detected and the loop restarts with another $T_{wait}$ period. If the timer is not zero, it has been started due to an earlier sw_input state change and the timer is decremented. When the timer changes from 1 to 0, the switch has been stable for $T_{mLS}$ and sw_state is updated to the value of sw_sample. The process continues indefinitely. If implemented in software (for example, in a microcontroller), execution of the algorithm loop could be started via a system timer that generates a System Interrupt after a period of $T_{wait}$ to cause one or more switches to be sampled. For a given value of $T_{mSL}$, larger timer values imply a shorter wait period (faster sampling) and the value of $T_{MSL}$ approaches that of $T_{mSL}$.

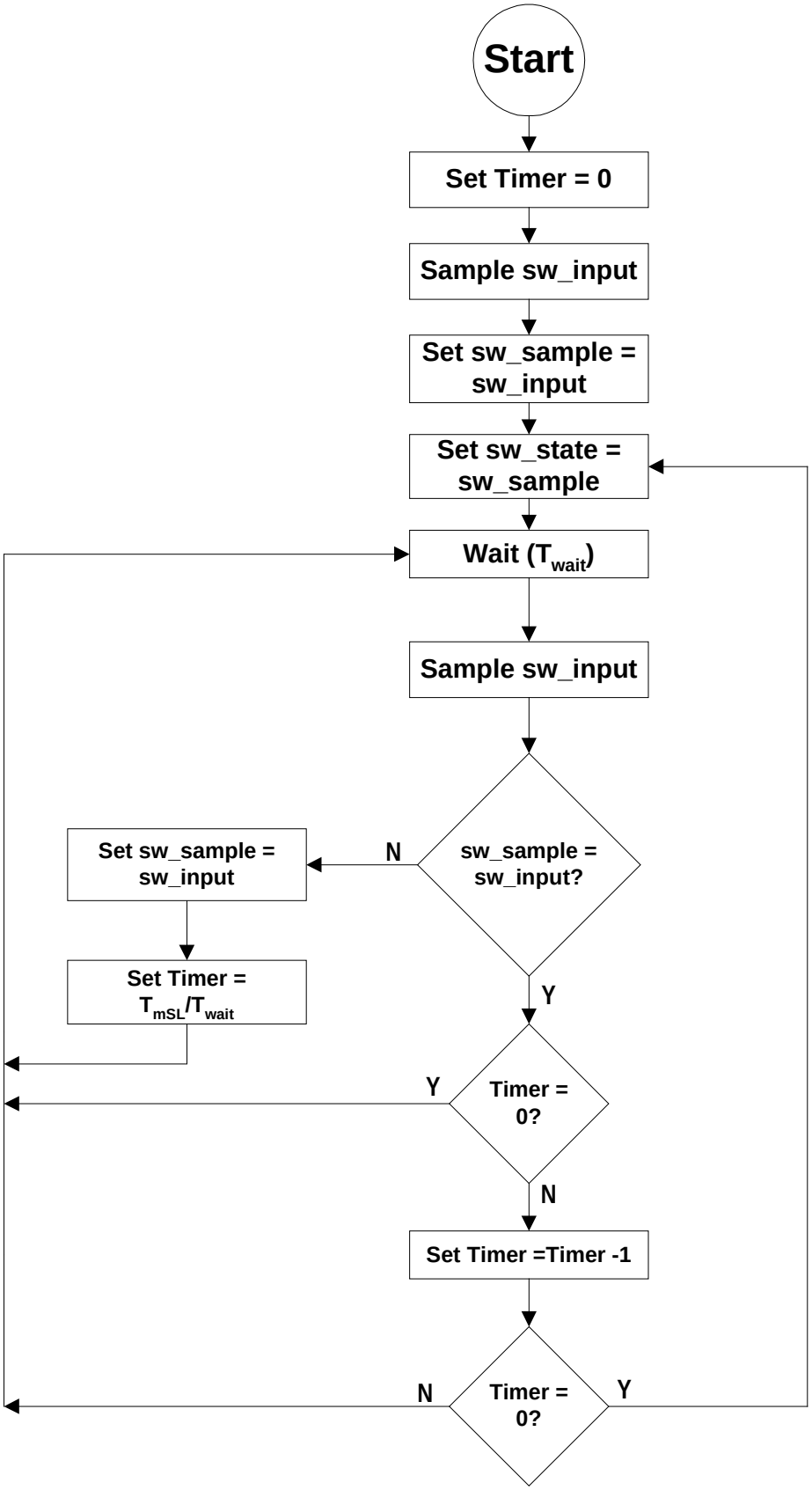



**Figure A-4: Digital Hardware or Software Debounce Algorithm**

Last, but not least, a very reliable technique for debouncing switches uses a latch and a SPDT switch (see Figure A-5). This technique relies on the fact that both contacts (throws) of the switch are never bouncing simultaneously. This SPDT switch characteristic is referred to as "break before make." Hence, after the switch changes mechanical state, the first bounce in the new state is used to change the state of an electronic latch and further bouncing on this contact has no effect on the electronic latch state.

This circuit has the lowest switch latency since the debounced state change occurs on the first bounce, contrasted with $T_{mSL}$ after the last bounce that is characteristic of time-domain filter debounce techniques. In addition, the switch bounce specification can be ignored since the circuit has no time-dependent elements that must be tuned to the bounce characteristics of the switch. However, this debounce approach does have caveats: SPDT switches are generally more expensive than SPST types, and two connections to the debounce circuit (pins) are required for each switch.

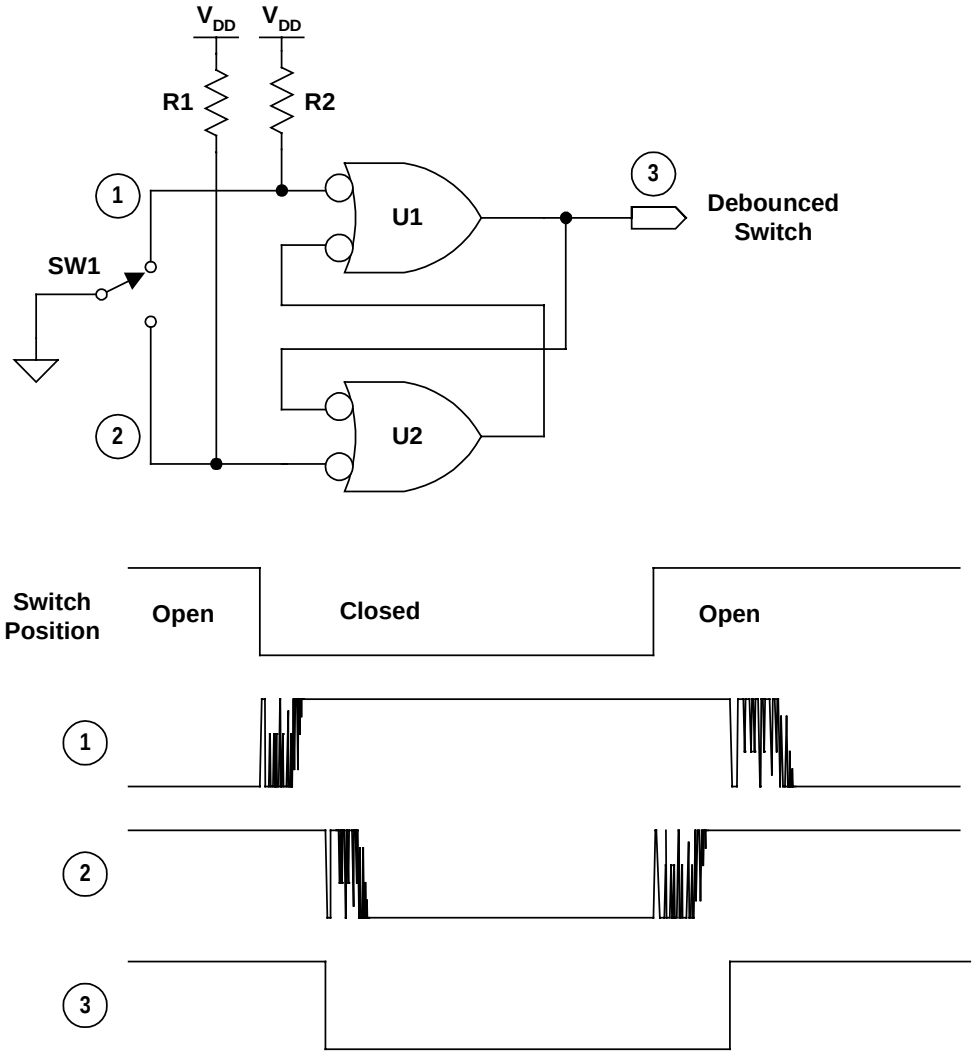



**Figure A-5: SPDT Switch Debounce Circuit**

# Appendix B. Compliance Checklist

This appendix provides design checklists for systems that are designed according to this specification. The requirements listed here are provided as an aid in designing and validating SHPC hardware and software. These lists are not comprehensive. In case of discrepancy between this checklist and the specification, the specification governs. SHPC hardware and software must meet all of the requirements in the specification, whether or not those requirements are repeated in this section.

## Firmware Requirements

## Power Controller Requirements

## Platform Requirements

## SHPC Requirements

## System Software Requirements